Culegere de probleme

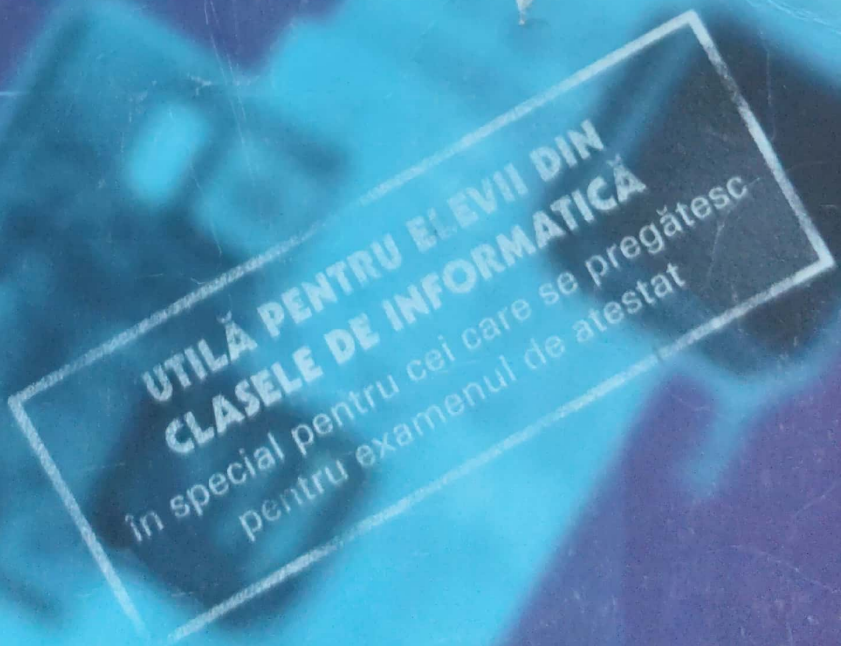
ALGORITMICA

BAZE DE DATE

MEDIUL WINDOWS

MICROSOFT WORD

DOS



COMPUTER LIBRIS
AGORA

Scanned with OKEN Scanner

Culegere de probleme

ALGORITMICĂ

BAZE DE DATE

MEDIUL WINDOWS

MICROSOFT WORD

DOS

COMPUTER LIBRIS
AGORA

Scanned with OKEN Scanner

Coordonator: Clara Ionescu

Această carte respectă programa şcolară
aprobată de Comisia Naţională de Specialitate Informatică
a Ministerului Educaţiei Naţionale
şi intrată în vigoare cu OM nr. 4701 din 16.08.1996

Autori:

Diana Gruiţă
Dana Lica
Dorin Mânz
Gina Plohod
Rodica Pintea
Doina Rancea
Adina Bălan
Luminiţa Condurache
Mihaela Giurgea
Eugen Ionescu
Ioan Maxim
Ioana Cucu
Diana Ignat
Emil Onea

Redactare: Clara şi Eugen Ionescu
Tehnoredactare computerizată: Dan Creţu
Control ştiinţific: Eugen Ionescu şi Dragoş Petraşcu
Coperta: Adrian Pop

Copyright ©1997 Computer Libris Agora

Cluj, Str. Universităţii nr.8/8
Tel: 064/192422

Bun de tipar: 8 martie 1998
Coli de tipar: 13
ISBN 973-97515-2-0
Tipar: Tipografia PALATINO, Cluj
Tel./Fax: 064/435701

Scanned with OKEN Scanner

Prefață

Oare chiar este nevoie de o culegere de probleme de tip "mozaic", ca aceasta?
Se pare că da!
Cine plânge după așa ceva?

În primul rând acei elevi "normali" care nu găsesc probleme de rezolvat în cărțile "șmechere" scrise pentru olimpici, dar au rezolvat toate temele pentru acasă. Dar, se bucură de această reuniune de probleme și liceanul din clasa a XII-a care ar vrea să se prezinte la atestatul de informatică și a auzit că la atestat va fi verificat și din algoritmică și din gestiunea bazelor de date și dintr-o serie de alte lucruri...

În această culegere veți găsi:

** probleme de algoritmică, majoritatea rezolvate în Pascal*
** probleme de gestiune a bazelor de date (abordate în FOX-PRO)*
** probleme care verifică îndemânarea de a lucra în mediul Windows și cu editorul Microsoft Word*
** probleme de verificare a cunoștințelor aferente sistemului de operare DOS*

Enunțurile pot constitui un punct de pornire pentru comisiile de acordare a atestatului de informatică la absolvirea liceului, deci constituie un sprijin pentru colectivele de profesori și pentru elevii care doresc să pregătească probleme pentru acest examen. De asemenea, culegerea poate fi folosită și în scopul pregătirii pe parcursul celor patru ani de liceu.

Culegerea a fost realizată de un număr mare de profesori cu o bogată experiență în predarea informaticii la toate nivelurile. Astfel, ea conține o mare varietate de enunțuri de probleme și rezolvări, abordând diferite metode și stiluri de programare.

* * *

Suntem convinși că o veți răsfoi cu interes și veți găsi multe amănunte, care vă vor folosi. De asemenea, știm că nu întotdeauna veți fi de acord cu stilul rezolvărilor prezentate, voi cunoscând altele mai elegante, mai performante. Dar tocmai acesta este scopul editurii Computer Libris Agora, de a vă ambiționa, de a vă face să fiți atenți și la ceilalți, de a deschide cât mai multe ferestre spre cunoștințele de informatică. Dacă vreți, considerați că participați la o întrecere și încercați să rezolvați aceste probleme (suntem convinși că puteți) și altfel, cu o eficiență în programare mai ridicată.

Cluj, 8 martie, 1998 *Editorul*

Scanned with OKEN Scanner

CUPRINS:

I. Algoritmi şi programare 5

II. Sisteme de gestiune a bazelor de date 101

III. Mediul Windows – Microsoft Word 200

IV. Sistemul de operare DOS 209

Scanned with OKEN Scanner

I. ALGORITMI ŞI PROGRAMARE

1. Cuvinte

Se consideră un şir de n litere mici distincte.

Să se genereze toate cuvintele care au următoarele proprietăţi:

– conţin exact m litere din şirul celor n litere date;

– literele din care sunt formate, apar în ordine lexicografică (crescătoare).

Datele de intrare se citesc de la tastatură. Cuvintele generate vor fi scrise în ordine lexicografică în fişierul de ieşire OUTPUT.TXT, un cuvânt pe fiecare linie.

Exemplu:

```
n=4                          OUTPUT.TXT:
m=2                          ab
Sirul de litere:             ac
a                            ad
b                            bc
c                            bd
d                            cd
```

Soluţie:

Pentru rezolvarea problemei se va utiliza metoda backtracking. Vom prezenta o implementare nerecursivă, pentru o mai uşoară înţelegere a ei. Literele se memorează în vectorul a, având componente de tip caracter. Stiva este reprezentată prin vectorul st, având componente de tip întreg, deoarece va reţine indicii literelor din vectorul a. Domeniul în care iau valori elementele vectorului st este intervalul [0,n]. Un nou nivel pe stivă se iniţializează prin depunerea valorii 0. Un nou element pe un nivel pe stivă se determină prin incrementarea valorii elementului curent. Un element depus pe stivă se validează prin verificarea condiţiei de ordine lexicografică a literelor din cadrul cuvântului. Am obţinut o soluţie în momentul în care nivelul stivei este egal cu numărul m impus pentru dimensiunea cuvintelor.

```
Program cuvinte;
type stiva=array[1..100] of Integer;
var st:stiva;
    a:array[1..1000] of Char;
    i,j,n,k,m:Integer;
    as,ev:Boolean;
    g:Text;

procedure succ(k:Integer; var st:stiva; var as:Boolean);
begin
  if st[k]<n then begin Inc(st[k]); as:=true end
           else as:=false
end;
```

Scanned with OKEN Scanner

```
procedure valid(k:Integer; st:stiva; var ev:Boolean);
begin
  ev:=true;
  if k>1
  then
    if a[st[k-1]]>=a[st[k]] then ev:=false
end;

Begin
  Write('n='); Readln(n);
  Write('m='); Readln(m);
  Writeln('Sirul de caractere:');
  for i:=1 to n do Readln(a[i]);
  Assign(g,'OUTPUT.TXT'); Rewrite(g);
  k:=1; st[k]:=0;
  while k>0 do
  begin
    repeat
      succ(k,st,as);
      if as then valid(k,st,ev)
    until not as or (as and ev);
    if as
    then
      if k=m
      then
        begin
          for i:=1 to m do Write(g,a[st[i]]);
          Writeln(g)
        end
      else
        begin Inc(k); st[k]:=0 end
    else Dec(k)
  end;
  Close(g)
End.
```

2. *Problema celor 8 regine*

Să se creeze un fişier cu tip, numit REGINE care va cuprinde soluţiile problemei celor 8 regine. Din fişierul standard de intrare se vor citi numerele de ordine ale soluţiilor care trebuie afişate. Se va afişa fiecare soluţie cerută, accesând fişierul REGINE în mod direct.

Reamintim: Problema cere determinarea tuturor modalităţilor de aşezare a 8 regine pe tabla de şah, astfel încât ele să nu se atace între ele.

Scanned with OKEN Scanner

Soluţie:

Prezentăm o implementare recursivă a metodei backtracking, utilizată pentru rezolvarea problemei.

Tabloul unidimensional `r` se utilizează pentru a păstra în elementele sale `r[i]` coloana în care se plasează a `i`-a regină.

Condiţiile de continuare constituie verificarea faptului dacă regina curentă atacă sau nu reginele aşezate deja. Se ştie că o regină atacă o altă regină dacă se află pe aceeaşi linie, sau pe aceeaşi coloană, respectiv pe aceeaşi diagonală cu aceasta. Din modul în care se dau valori elementelor şirului `r` rezultă că nu trebuie făcute verificări pentru atacul pe linie, deoarece la pasul `i` se plasează o singură regină (a `i`-a) pe a `i`-a linie. Rezultă că în funcţia `nu_se_ataca` se verifică doar atacul pe coloane şi diagonale.

Obţinem o soluţie în momentul în care s-a plasat şi a `8`-a regină. Se opreşte căutarea soluţiilor atunci când ar trebui căutată o poziţie nouă pentru regina de pe linia `1` dincolo de coloana `8`.

```
Program regine;
type sol=array[1..8] of Byte;
var f:file of sol;
    r:sol; { r[i]=coloana in care se afla regina din linia i }
    n:Longint;          { numarul de ordine al solutiei cerute }

function nu_se_ataca(i:Byte):Boolean;
var j:Byte;
begin
  nu_se_ataca:=True;
  for j:=1 to i-1 do
    if (r[i]=r[j]) or (i-j=Abs(r[i]-r[j]))
    then begin nu_se_ataca:=False;
               Exit
         end
end;

procedure regina(i:Byte);
var j:Byte;
begin
  for j:=1 to 8 do
  begin
    r[i]:=j;
    if nu_se_ataca(i)
    then
      if i=8 then Write(f,r)          {scrie solutia in fisier}
             else regina(i+1)
  end
end;
```

```
procedure scrie;
var i,j:Byte;
begin
  for i:=1 to 8 do
  begin
    for j:=1 to 8 do
      if r[i]=j then Write(' R ')
                else Write(' . ');
    Writeln
  end
end;

Begin
  Assign(f,'REGINE'); Rewrite(f);
  regina(1);
  Close(f);
  Reset(f);
  Write('Numarul de ordine al solutiei(1-92):');
  While not Eof do
  begin
    Readln(n);
    if n in [1..92]
    then begin
           Seek(f,n-1); Read(f,r);
           scrie
         end
    else Writeln('Numarul de ordine trebuie sa fie in [1,92]');
    Write('Numarul de ordine al solutiei(1-92):')
  end;
  Close(f)
End.
```

3. *Labirint*

Fişierul text `LABIRINT.TXT`, conţine pe prima linie dimensiunile `m` şi `n` ale labirintului, iar pe următoarele linii sunt elementele tabloului bidimensional corespunzător labirintului. Culoarele labirintului sunt codificate prin valori egale cu `1`, zidurile cu `0`. O persoană trebuie să ajungă din punctul corespunzător coordonatelor `(xi,yi)` în poziţia `(xf,yf)`, traversând labirintul pe orizontală şi pe verticală.

Din fişierul standard de intrare se vor citi coordonatele punctului iniţial `(xi,yi)` şi coordonatele punctului final `(xf,yf)` ale traseului.

Să se creeze un fişier text `TRASEE.TXT` cu toate traseele prin care se poate ajunge din punctul `(xi,yi)` în `(xf,yf)`. Un traseu nu va conţine aceeaşi poziţie de două ori.

Să se afişeze conţinutul fişierului pe ecran.

Scanned with OKEN Scanner

Soluţie:

Labirintul se păstrează în tabloul `lab`, iar soluţiile cu traseele din labirint se scriu în tabloul `sol`. Determinarea drumurilor se realizează cu metoda backtracking. Prezentăm o implementare recursivă.

Tabloul `sol` se iniţializează cu `0` şi pe parcursul căutării traseului din labirint, acesta se marchează cu valoarea pasului (păstrată în variabila `pas`). Pentru a decide dacă un "pas" din traseu este corect ales sau nu, trebuie verificată valoarea din tabloul `lab` (o valoare diferită de `1` semnifică faptul că în poziţia respectivă este zid) şi valoarea din tabloul `sol` (o valoare diferită de `0` semnificând faptul că poziţia respectivă face parte deja din traseu). În cazul îndeplinirii condiţiilor de continuare se efectuează pasul şi se verifică dacă s-a ajuns în poziţia finală, caz în care se poate afişa o soluţie.

Dintr-o anumită poziţie `(i,j)`, traseul se poate continua în patru direcţii: în liniile `i-1` şi `i+1`, respectiv coloanele `j-1` şi `j+1`. După tipărirea unei soluţii, respectiv abandonarea unei ramificaţii, elementul curent din tabloul `sol` se reiniţializează cu `0` (se şterge din traseu).

```
Program trasee_in_labirint;
type tablou=array[0..10,0..20] of Byte;
var f1,f2:Text;
    sol,lab:tablou;
    xi,xf,yi,yf,m,n,k:Byte;

procedure citire(var f:Text; var x:tablou);
var i,j:Byte;
begin
  for i:=1 to m do
  begin
    for j:=1 to n do
      Read(f,x[i,j]);
    Readln(f)
  end
end;

procedure scriere(var f:Text);
var i,j:Byte;
begin
  for i:=1 to m do
  begin
    for j:=1 to n do
      Write(f,sol[i,j]:3);
    Writeln(f)
  end
end;
```

```
procedure labirint(i,j,pas:Byte);
begin
  if (lab[i,j]=1) and (sol[i,j]=0)
  then
    begin
      sol[i,j]:=pas;
      if (i=xf) and (j=yf) then scriere(f2)
      else
        begin
          labirint(i-1,j,pas+1);
          labirint(i+1,j,pas+1);
          labirint(i,j-1,pas+1);
          labirint(i,j+1,pas+1)
        end;
      sol[i,j]:=0
  end
end;

Begin
  Assign(f1,'LABIRINT.TXT'); Reset(f1);
  Assign(f2,'TRASEE.TXT'); Rewrite(f2);
  Readln(f1,m,n); Citire(f1,lab);
  Close(f1);
  repeat
    Write('Coord, initiale si finale:');
    Readln(xi,yi,xf,yf)
  until ([xi,xf]<=[1..m]) and ([yi,yf]<=[1..n]);
  labirint(xi,yi,1);
  Close(f2);
  Reset(f2);
  while not Eof(f2) do
  begin citire(f2,sol); Inc(k);
        Writeln('solutia ',k,':'); scriere(Output)
  end;
  Close(f2)
End.
```

4. Produs cartezian

Se citeşte din fişierul standard de intrare un număr $n \in N^*$ şi o mulţime M de litere mici din alfabetul englezesc.

Să se determine şi să se scrie în fişierul text CART elementele produsului cartezian $M \times M \times \ldots \times M = M^n$.
(de n ori)

Să se listeze conţinutul acestui fişier.

Scanned with OKEN Scanner

Soluţie:

```
Program produs_cartezian;
var  n:Byte;
     m:string;
     x:array[1..20] of Byte;
     f:Text;

procedure scrie;
var i:Byte;
begin
  for i:=1 to n do Write(f,m[x[i]]);
  Writeln(f)
end;

procedure cart(i:Byte);
var j:Byte;
begin
  for j:=1 to Length(m) do
  begin
    x[i]:=j;
    if i=n then scrie
           else cart(i+1)
  end
end;

Begin
  Write('n='); Readln(n);
  Write('multimea M:'); Readln(m);
  Assign(f,'CART'); Rewrite(f);
  cart(1);
  Close(f);
  Reset(f);
  while not Eof(f) do
  begin
    Readln(f,m);
    Write(m); Readln
  end;
  Close(f)
End.
```

5. Submulţimi (1)

Se citeşte o linie din fişierul standard de intrare care cuprinde o mulţime de numere întregi. Să se genereze submulţimile acestei mulţimi, memorându-le în câte o linie a fişierului text SUBM.TXT.

Să se afişeze fişierul creat.

Scanned with OKEN Scanner

Soluţie:

```
Program submultimi;
var  n:Byte;
     m:array[1..20] of Integer;
     x:array[0..20] of Byte;
     f:Text;

procedure scrie(i:Byte);
var j:Byte;
begin
  for j:=1 to i do Write(f,' ',m[x[j]]);
  Writeln(f)
end;

procedure subm(i:Byte);
var j:Byte;
begin
  for j:=x[i-1]+1 to n do
  begin
    x[i]:=j;
    scrie(i);
    subm(i+1)
  end
end;

Begin
  n:=0;
  Write('Multimea M:');
  while not Eoln do
  begin Inc(n); Read(m[n]) end;
  Readln;
  Assign(f,'SUBM.TXT'); Rewrite(f);
  subm(1);
  Close(f); Reset(f);
  Writeln('Submultimile:');
  Writeln(Chr(237):3);
  while not Eof(f) do
  begin
    while not Eoln(f) do
    begin
      Read(f,n); Write(n:3)
    end;
    Readln; Readln(f)
  end;
  Close(f)
End.
```

Scanned with OKEN Scanner

6. *Submulţimi (2)*

Se consideră o mulţime formată din n elemente numere întregi.

Să se genereze toate submulţimile acestei mulţimi având proprietatea că suma elementelor lor este egală cu s.

Datele de intrare se citesc de la tastatură.

Elementele submulţimilor vor fi scrise pe câte o linie în fişierul OUTPUT.TXT; ultima linie va conţine numărul de submulţimi.

Exemplu:

```
n=5
s=4
Sirul de numere: 1 3 2 -1 5
```

```
OUTPUT.TXT:
1 3
3 2 -1
-1 5
3
```

Soluţie:

Pentru rezolvarea problemei se va utiliza metoda backtracking. Prezentăm o implementare nerecursivă. Numerele date se vor păstra în vectorul a. Stiva este reprezentată prin vectorul st, având componente de tip întreg. Deoarece acestea sunt indicii elementelor din vectorul a, ele iau valori din intervalul [0,n].

Un nou nivel pe stivă se iniţializează prin depunerea valorii 0. Un nou element pe un nivel pe stivă se determină prin incrementarea valorii elementului curent. Validarea unui element depus în stivă constă în verificarea condiţiei ca un element să nu fi fost depus anterior. De asemenea, pentru evitarea repetării soluţiilor (datorită comutativităţii adunării) numerele se vor depune pe stivă în ordine crescătoare.

Considerăm că am obţinut o soluţie dacă suma valorilor depuse pe stivă este egală cu numărul s dat la intrare.

```
Program submultimi;
type stiva=array[1..100] of Integer;
var st,a:stiva;
    i,j,n,k,s,nr:Integer;
    as,ev,esol:Boolean;
    g:Text;
procedure succ(k:Integer; var st:stiva; var as:Boolean);
begin
  if st[k]<n
  then begin
         st[k]:=st[k]+1;
         as:=true
       end
  else as:=false
end;
```

Scanned with OKEN Scanner

```
procedure valid(k:Integer; st:stiva; var ev:Boolean);
begin
  ev:=true;
  if k>1 then if st[k]<=st[k-1] then ev:=false;
  for i:=1 to k-1 do if st[k]=st[i] then ev:=false
end;
procedure solutie(k:Integer; var esol:Boolean);
var su:Integer;
begin
  su:=0;
  for i:=1 to k do su:=su+a[st[i]];
  esol:=su=s
end;
procedure tipar(k:Integer; st:stiva);
var t:Integer;
begin
  for t:=1 to k do Write(g,a[st[t]],' '); Writeln(g);
  Inc(nr)
end;
Begin
  Write('n='); Readln(n);
  Write('s='); Readln(s);
  Assign(g,'OUTPUT.TXT'); Rewrite(g);
  Write('Sirul de numere:');
  for i:=1 to n do Read(a[i]);
  k:=1; st[k]:=0; nr:=0;
  while k>0 do
  begin
    repeat
      succ(k,st,as);
      if as then valid(k,st,ev)
    until not as or (as and ev);
    if as
    then
      begin
        solutie(k,esol);
        if esol then tipar(k,st)
                else begin Inc(k); st[k]:=0 end
      end
    else Dec(k)
  end;
  Write(g,nr);
  Close(g)
End.
```

Scanned with OKEN Scanner

7. *Generare de funcţii*

Se citesc din fişierul standard de intrare două mulţimi de numere întregi A şi B (date fiecare pe câte o linie).

Să se genereze toate funcţiile f:A→B şi să se memoreze în fişierul text FUNCTII (o funcţie pe o linie).

Să se afişeze fişierul creat.

Soluţie:

Prezentăm o implementare iterativă a metodei backtracking. Observăm că funcţiile cerute sunt determinate de şirul imaginilor elementelor a[1],a[2],...,a[n]. Pornim cu configuraţia: b[1],b[1],...,b[1], apoi mărim ultimul indice rând pe rând la valorile 2,3,...,m, după care se măreşte penultimul indice cu 1, ultimul luând din nou toate valorile de la 1 până la m; apoi se măreşte din nou penultimul indice, ş.a.m.d., generarea terminându-se după obţinerea configuraţiei b[m],b[m],...,b[m].

```
Program generare_functii;
var a,b:array[1..20] of Integer;
    f:array[1..10] of Byte;
    n,m,i,j:Byte;
    fo:Text;

Begin
  Assign(fo,'FUNCTII'); Rewrite(fo);
  n:=0;
  while not Eoln do begin Inc(n); Read(a[n]) end;
  Readln;
  m:=0;
  while not Eoln do begin Inc(m); Read(b[m]) end;
  for i:=1 to n do
  begin Write(fo,a[i]:3); f[i]:=1 end;
  for i:=1 to n do Write(fo,b[1]:3);
  Writeln(fo);
  i:=n;
  repeat
    while f[i]<m do
    begin
      f[i]:=f[i]+1;
      for j:=1 to n do Write(fo,a[j]:3);
      for j:=1 to n do Write(fo,b[f[j]]:3);
      i:=n; Writeln(fo)
    end;
    f[i]:=1; i:=i-1
  until i=0;
  Close(fo);
```

Scanned with OKEN Scanner

```
  Reset(fo);
  while not Eof(fo) do
  begin
    Writeln; Write('   x:');
    for i:=1 to n do Read(fo,a[i]);
    for i:=1 to n do Write(a[i]:3);
    Writeln; Write('f(x):');
    for i:=1 to n do Read(fo,b[i]); Readln(fo);
    for i:=1 to n do Write(b[i]:3)
  end;
  Writeln; Readln
End.
```

8. *Listă de cuvinte*

Se consideră o secvență formată din n cuvinte.

Să se construiască cu ele cel mai lung şir în care fiecare cuvânt are ultima literă mai mică în sens lexicografic decât prima literă a cuvântului următor.

Pe prima linie a fişierului de intrare INPUT.TXT este scris numărul n; pe următoarele n linii sunt scrise cuvintele c1, c2, ..., cn.

Şirul maximal va fi scris în fişierul de ieşire OUTPUT.TXT. În cazul în care există mai multe soluții se va scrie doar una.

Exemplu:

```
INPUT.TXT:          OUTPUT.TXT:
5                   atuncirosuvine
atunci
cand
rosu
are
vine
```

Soluție:

Pentru rezolvarea problemei se va utiliza metoda backtracking. Prezentăm o implementare iterativă (nerecursivă).

Cuvintele citite formează vectorul a, având componente de tip **string**. Stiva este reprezentată prin vectorul st, având elemente de tip întreg, reprezentând indicii cuvintelor din vectorul a. Astfel, elementele vectorului st iau valori din intervalul [0,n]. Un nou nivel pe stivă se inițializează prin depunerea valorii 0. Un nou element pe un nivel pe stivă se determină prin incrementarea valorii elementului curent. Condiția de continuare constă în verificarea de ordine lexicografică între ultima literă a cuvântului existent pe nivelul anterior şi prima literă a cuvântului depus pe nivelul curent. De asemenea, se impune condiția ca elementele din st să fie distincte. Am obținut o soluție în momentul în care nivelul stivei este egal cu n.

Scanned with OKEN Scanner

Evident, dacă cele n cuvinte pot fi alipite în condiţiile impuse, atunci şirul format este de lungime maximă. Dacă nu există această posibilitate, variabila max va fi actualizată mereu cu cel mai lung şir obţinut, iar vectorul st1 va păstra configuraţia stivei care a condus la această lungime.

```
Program sir_cuvinte;
type stiva=array[1..100] of Integer;
var st,st1:stiva;
    a:array[1..1000] of string[30];
    i,j,n,k,m,max,k1:Integer;
    as,ev,esol:Boolean;
    g,f:Text;

procedure succ(k:Integer; var st:stiva; var as:Boolean);
begin
  if st[k]<n then begin Inc(st[k]); as:=true end
            else as:=false
end;

procedure valid(k:Integer; st:stiva; var ev:Boolean);
var c:Integer;
begin
  ev:=true;
  if k>1
  then
    begin
      c:=Length(a[st[k-1]]);
      if a[st[k-1]][c]>a[st[k]][1] then ev:=false
    end;
  for i:=1 to k-1 do
    if st[k]=st[i] then ev:=false
end;

procedure solutie(k:Integer; var esol:Boolean);
var inf:Integer;
begin
  inf:=0;
  for i:=1 to k do inf:=inf+Length(a[i]);
  if inf>max
  then
    begin
      for i:=1 to k do st1[i]:=st[i];
      max:=inf; k1:=k
    end;
  esol:=k=n
end;
```

Scanned with OKEN Scanner

```
Begin
  Assign(f,'INPUT.TXT'); Reset(f); Readln(f,n);
  for i:=1 to n do Readln(f,a[i]);
  max:=0; k:=1; st[k]:=0;
  while k>0 do
  begin
    repeat
      succ(k,st,as);
      if as then valid(k,st,ev)
    until not as or (as and ev);
    if as
    then begin
           solutie(k,esol);
           if not esol
           then begin Inc(k); st[k]:=0 end
         end
    else Dec(k)
  end;
  Assign(g,'OUTPUT.TXT'); Rewrite(g);
  for i:=1 to k1 do Write(g,a[st1[i]]);
  Close(g)
End.
```

9. Străzi

Într-un oraş există n centre comerciale unite printr-o reţea stradală, toate fiind cu circulaţie în sens unic.

Se cere determinarea unui centru în care se poate ajunge din oricare centru.

Pe prima linie a fişierului INPUT.TXT este scris numărul de centre (n); pe următoarele linii sunt scrise perechi de numere: i j – reprezentând strada între centrele i şi j cu sensul de parcurgere de la i la j.

Rezultatul se va afişa pe ecran; în cazul în care există mai multe soluţii se vor afişa toate.

Exemplu:

INPUT.TXT:	Pe ecran se va afişa:
7	3
1 2	
2 7	
1 7	
7 3	
5 6	
5 4	
6 4	
4 3	

Scanned with OKEN Scanner

Soluţie:

Pentru reprezentarea centrelor şi străzilor folosim un graf orientat, reprezentat prin matricea de adiacenţă a_{nxn}. Centrele care respectă condiţiile problemei se identifică în modul următor: se determină, cu ajutorul algoritmului *Roy-Warshall*, matricea drumurilor (închiderea tranzitivă). Apoi, însumând pe fiecare coloană `j` elemente, vom determina numărul de noduri prin care se poate ajunge în nodul `j`. Evident, constituie soluţie acele noduri care au suma pe coloane egală cu `n-1` (fără elementul `a[j,j]`).

```
Program graf;
var a:array[1..100,1..100] of Integer;
    i,j,n,k:Integer;
    f:Text;

procedure citeste;
begin
  for i:=1 to n do
    for j:=1 to n do a[i,j]:=0;
  Assign(f,'INPUT.TXT'); Reset(f); Readln(f,n);
  while not Eof(f) do
  begin Readln(f,i,j); a[i,j]:=1 end
end;

procedure Roy_Warshall;
begin
  for k:=1 to n do
    for i:=1 to n do
      for j:=1 to n do
        if a[i,j]=0 then a[i,j]:=a[i,k]*a[k,j]
end;

procedure suma;
var s:Integer;
begin
  for j:=1 to n do
  begin
    s:=0;
    for i:=1 to n do if i<>j then s:=s+a[i,j];
    if s=n-1 then Writeln(j)
  end
end;
Begin
  citeste;
  Roy_Warshall;
  suma
End.
```

10. Maximizări

Se consideră un graf $G=(X,U)$ neorientat, neconex.

Vom considera că două noduri sunt *conectate* dacă există un lanț care le unește.

Să se determine o pereche de noduri între care nu există muchie, astfel încât, prin adăugarea muchiei corespunzătoare, numărul de noduri conectate să fie maxim.

Pe prima linie a fișierului `INPUT.TXT` este scris numărul de noduri (`n`); pe următoarele linii sunt scrise perechi de numere: `i j` – reprezentând perechi de noduri adiacente. Rezultatul se va afișa pe ecran; în cazul în care există mai multe soluții se va afișa una singură.

Exemplu:

`INPUT.TXT:`	Pe ecran se va afișa:
`9`	`1 5`
`1 2`	
`1 3`	
`2 4`	
`3 4`	
`5 6`	
`5 7`	
`6 7`	
`8 9`	

Soluție:

Reprezentăm graful neorientat prin matricea $a_{2\times m}$ a muchiilor. Numărul `m` al muchiilor este determinat la citire. Se identifică muchia ce trebuie adăugată determinând toate componentele conexe (procedura `conexitate`) care va reține în vectoul `L` elemente cu semnificația `L[i]=j` dacă nodul `i` face parte din componenta numărului `j`. Apoi se determină primele două componente cu număr maxim de noduri. Muchia dorită se obține unind două noduri din cele două componente determinate.

```
Program conex;
var a:array[1..2,1..100] of Integer;
    L:array[1..100] of Integer;
    i,j,n,m,c,c1:Integer;
    f:Text;

procedure citeste;
begin
  Assign(f,'INPUT.TXT'); Reset(f); Readln(f,n);
  m:=0;
  while not Eof(f) do
  begin Readln(f,i,j);
        Inc(m); a[1,m]:=i; a[2,m]:=j
  end
end;
```

Scanned with OKEN Scanner

```
procedure conexitate;
var p,s,q,max,ind,nr,k,v:Integer;
begin
  for i:=1 to n do L[i]:=i;
  for i:=1 to m do
  begin
    p:=a[1,i];
    q:=a[2,i];
    if L[p]<L[q]
    then
      for j:=1 to n do
        if L[j]=L[q] then L[j]:=L[p]
    else
      for j:=1 to n do
        if L[j]=L[p] then L[j]:=L[q]
  end;
  for k:=1 to 2 do
  begin
    max:=0;
    for i:=1 to n do
    begin
      if L[i]<>0
      then
        begin
          nr:=0; p:=L[i];
          for j:=1 to n do
            if L[j]=p then Inc(nr);
          if max<nr
          then
            begin
              max:=nr;
              ind:=i;
              v:=L[i]
            end
        end
    end;
    Write(ind,' ');
    for i:=1 to n do
      if L[i]=v then L[i]:=0
  end
end;

Begin
  citeste;
  conexitate
End.
```

Scanned with OKEN Scanner

11. Numere mari

Se consideră două numere naturale `n` şi `m` care au cel puţin `200` de cifre. Să se determine produsul celor două numere.

Pe prima linie a fişierului de intrare `INPUT.TXT` este scris numărul `n`, având cifrele despărţite prin câte un spaţiu; pe a doua linie este scris numărul `m`, în mod similar.

În fişierul `OUTPUT.TXT` se va scrie pe fiecare linie câte o cifră din rezultat.

Soluţie:

Problema se rezolvă simulând înmulţirea prin calcul pe vectori de cifre. Numerele sunt reţinute în vectorii `a` şi `b` în care fiecare element constituie o cifră. Produsul se determină în vectorul `c` a cărui dimensiune maximă este dată de constanta `llimita`.

Cifrele lui `c` se determină începând cu `c[llimita]` şi terminând cu `c[1]` prin înmulţirea cifrelor corespunzătoare din `a`, respectiv din `b`. Evident, înmulţirea celor două cifre poate produce o cifră de transport, care trebuie gestionată corespunzător. La afişare se vor ignora zerourile nesemnificative.

```
Program nr_mari;
const limita=10000;
      llimita=20000;
var a,b:array [1..limita] of Byte;
    c:array [1..llimita] of Byte;
    f,g:Text;
    rez,adun,n,m,poz,pz,i,j,ra,ri,v:Integer;

Begin
  Assign(f,'INPUT.TXT'); Reset(f);
  Assign(g,'OUTPUT.TXT'); Rewrite(g);
  n:=0;
  while not Seekeoln(f) do
  begin
    n:=n+1;
    Read(f,a[n])
  end;
  Readln(f);
  m:=0;
  while not Seekeoln(f) do
  begin
    m:=m+1;
    Read(f,b[m])
  end;
  Close(f);
  for i:=1 to llimita do c[i]:=0;
  poz:=llimita;
```

Scanned with OKEN Scanner

```
  for i:=m downto 1 do
  begin
    pz:=poz; ra:=0; ri:=0;
    for j:=n downto 1 do
    begin
      v:=b[i]*a[j]+ri;
      adun:=v mod 10;
      ri:=v div 10;
      rez:=adun+c[pz]+ra;
      ra:=rez div 10;
      c[pz]:=rez mod 10;
      pz:=pz-1
    end;
    ri:=ri+ra;
    while ri>0 do
    begin
      adun:=c[pz]+ri;
      c[pz]:=adun mod 10;
      ri:=adun div 10
    end;
    poz:=poz-1
  end;
  poz:=1;
  while c[poz]=0 do poz:=poz+1;
  for i:=poz to llimita do Writeln(g,c[i]);
  Close(g)
End.
```

12. *Conjectura lui Goldbach*

Se consideră un număr n>6, par. Să se determine toate reprezentările lui n ca sumă de numere prime, sumă cu număr minim de termeni.

Rezultatele vor fi scrise în fişierul OUTPUT.TXT, fiecare linie conţinând toţi termenii dintr-o reprezentare.

Exemplu:

```
n=8          OUTPUT.TXT:
             1 7
             3 5
```

Soluţie:

Rezolvarea problemei se bazează pe faptul că reprezentarea unui număr par ca sumă formată din număr minim de termeni primi este dată de conjectura Goldbach: numărul se poate reprezenta ca sumă de două numere prime. În aceste condiţii se verifică numerele i (de la 1 la jumătatea numărului dat n), în scopul de a determina numerele care sunt prime, apoi se verifică dacă şi numărul n-i este prim.

Scanned with OKEN Scanner

```
Program termeni;
var n:Integer;
    sw:Boolean;
    g:Text;

procedure prime(a:Integer; var sw:Boolean);
var i,x:Integer;
begin
  sw:=true; i:=2;
  x:=Trunc(Sqrt(a));
  while sw and (i<=x) do
    if a mod i=0 then sw:=false
                 else Inc(i)
end;

procedure Goldbach;
var i:Integer;
begin
  for i:=1 to Trunc(n/2) do
  begin
    prime(i,sw);
    if sw then prime(n-i,sw);
    if sw then Writeln(g,i,' ',n-i)
  end;
  Close(g)
end;

Begin
  Readln(n);
  Assign(g,'OUTPUT.TXT');
  Rewrite(g);
  Goldbach
End.
```

13. Factor prim

Se consideră un şir de n numere întregi. Să se determine factorul prim care apare la puterea cea mai mare în descompunerea produsului celor n numere.

Pe prima linie a fişierului INPUT.TXT este scris numărul n, reprezentând numărul de elemente din şir; pe următoarea linie sunt scrise cele n numere.

Răspunsul se va afişa pe ecran sub forma:

i^j – cu semnificaţia: factorul i la puterea j

Exemplu:

INPUT.TXT:	Pe ecran se va afişa:
4	3^4
24 9 17 21	

Scanned with OKEN Scanner

Soluţie:

```
Program Factor_prim;
const InputFile='INPUT.TXT'; { fisierul cu datele de intrare }
type vector=array[1..100] of Integer;

procedure Citiri(s:string; var n:Integer; var X:vector);
var i:Integer;          { citeste din fisierul text cu numele s }
    f:Text;             { vectorul X cu n componente }
begin
  Assign(f,s); Reset(f);
  Readln(f,n);
  for i:=1 to n do Read(f,X[i]);
  Readln(f);
  Close(f)
end;

function prim(n:Integer):Boolean;
var d,i:Integer;                { intoarce true daca n este prim }
begin                           { si false daca nu este prim     }
  prim:=false;
  if n<2 then Exit;
  for i:=2 to Round(Sqrt(n)) do
    if n mod i=0 then Exit;
  prim:=true
end;

function putere1(d,n:Integer):Integer;
var rez:Integer;          { intoarce puterea la care apare d in }
begin                     { in descompunerea in factori a lui n }
  rez:=0;
  while n mod d=0 do
  begin
    Inc(rez);
    n:=n div d
  end;
  putere1:=rez
end;

function putere(nr:Integer; n:Integer; X:vector):Integer;
var i,rez:Integer;          { intoarce puterea                    }
begin                       { la care apare nr in X[1]*...*X[n] }
  rez:=0;
  for i:=1 to n do
    Inc(rez,putere1(nr,X[i]));
  putere:=rez
end;
```

```
procedure DivExpMax(n:Integer;X:vector; var divizor,emax:Integer);
var d,e,i:Integer;
begin
  divizor:=1;
  emax:=0;
  for i:=1 to n do
  begin
    for d:=2 to (X[i] div 2) do
      if prim(d)
      then
        begin
          e:=putere(d,n,X);
          if e>emax
          then
            begin
              emax:=e;
              divizor:=d
            end
        end;
    if prim(X[i])
    then
      begin
        e:=putere(X[i],n,X);
        if e>emax
        then
          begin
            emax:=e;
            divizor:=X[i]
          end
      end
  end
end;

var n:Integer; X:vector;                    { modulul principal }
    divizor,emax:Integer;

Begin
  Citiri(InputFile,n,X);
  DivExpMax(n,X,divizor,emax);
  Writeln(divizor,'^',emax)
End.
```

Scanned with OKEN Scanner

14. Descompunere în factori primi

Să se citească din fişierul standard de intrare un număr neprecizat de numere naturale n. Să se afişeze pentru fiecare număr citit două linii:
– pe prima linie se va scrie valoarea lui n;
– pe a doua linie se vor scrie factorii primi şi puterile lor în descompunerea lui n în factori primi.

Să se citească fişierul creat şi să se afişeze pe ecran descompunerea fiecărui număr.

Soluţie:

```
Program factori_primi;
var nn,n,d:0..maxlongint;
    k:Byte;              { numara de cate ori se imparte n la d }
    divizor:Boolean;     { semnaleaza existenta/absenta a cel }
                         { putin un divizor (factor)          }
Begin
  Writeln('Introduceti numerele pe o linie!');
  while not Eoln do
  begin
    Read(n);
    k:=0;
    d:=2;
    divizor:=false;
    Writeln;
    Write(n,'=');
    nn:=n;               { avem nevoie de n nealterat in while }
    while d<=nn div 2 do
    begin
      while n mod d = 0 do
      begin
        k:=k+1;
        n:=n div d;
        divizor:=true
      end;
      if k>1 then Write(d,'^',k,'*'){ exista factor la putere>1 }
             else
               if k=1 then Write(d,'*');  { factor la putere=1 }
      k:=0;
      Inc(d)
    end;
    if not divizor then Write(nn,'*');
    n:=nn;
    Write(#8,' '){ suprascriem cu blanc ultimul * scris in plus }
  end
End.
```

Scanned with OKEN Scanner

15. Cel mai mare divizor comun

Se consideră n numere scrise în baza 16.

Să se determine cel mai mare divizor comun al celor n numere.

Pe prima linie a fişierului INPUT.TXT este scris numărul de numere n; pe următoarele n linii se găsesc numerele scrise în hexazecimal.

Numărul care reprezintă cel mai mare divizor comun al celor n numere, se va afişa pe ecran.

Exemplu:

INPUT.TXT:	Se va afişa:
4	2
1A	
8	
10	
2	

Soluţie:

Odată cu citirea datelor de intrare se realizează şi conversia numerelor din baza 16 în baza 10. Numerele astfel obţinute vor fi memorate în vectorul a, apoi se determină c.m.m.d.c. prin scăderi succesive.

```
Program c_m_m_d_c;
var a:array[1..1000] of Integer;
    b:string;
    i,j,c,n:Integer;
    f:Text;
Begin
  Assign(f,'INPUT.TXT'); Reset(f); Readln(f,n);
  for i:=1 to n do
  begin
    Readln(f,b); a[i]:=0;
    for j:=1 to Length(b) do
    begin
      if b[j] in ['a'..'f'] then b[j]:=Upcase(b[j]);
      case b[j] of
        'A':a[i]:=a[i]*16+10;
        'B':a[i]:=a[i]*16+11;
        'C':a[i]:=a[i]*16+12;
        'D':a[i]:=a[i]*16+13;
        'E':a[i]:=a[i]*16+14;
        'F':a[i]:=a[i]*16+15;
        else a[i]:=a[i]*16+Ord(b[j])-Ord('0')
      end
    end
  end;
```

Scanned with OKEN Scanner

```
  Close(f);
  c:=a[1];
  for i:=2 to n do
    while c<>a[i] do
      if c>a[i] then c:=c-a[i]
                else a[i]:=a[i]-c;
  Writeln(c)
End.
```

16. Anagrame

Se consideră o matrice pătratică de dimensiune `n` (`n<100`) ale cărei elemente sunt cuvinte formate din cel mult `10` litere mici.

Se cere:

a) Să se determine cuvântul de lungime minimă din matrice. În cazul în care există mai multe soluţii se va afişa una.

b) Dându-se un cuvânt, să se verifice dacă există vreun element al matricei care este anagrama lui. Dacă există mai multe soluţii se vor afişa toate.

Pe prima linie a fişierului `INPUT.TXT` este scris `n` (numărul de linii şi coloane al matricei); pe următoarele n^2 linii este scris câte un cuvânt; pe ultima linie este scris cuvântul căruia i se caută anagrama.

Răspunsurile se vor afişa pe ecran.

Exemplu:

`INPUT.TXT:`	Pe ecran se va afişa:
`2`	`cu`
`are`	`cuvantul care este anagrama lui cera`
`care`	
`peste`	
`cu`	
`cera`	

Soluţie:

Se observă că pentru rezolvarea problemei nu este nevoie să se păstreze cuvintele într-o matrice, de aceea acestea s-au citit în vectorul `a`.

Prima cerinţă a problemei are o rezolvare imediată, determinându-se în procedura `minim` indicele `poz` pentru care cuvântul `a[poz]` are lungimea minimă.

A doua cerinţă este rezolvată în procedura `anagrame` unde sunt verificate doar acele cuvinte din `a` care au aceeaşi lungime ca şi cuvântul `an` căruia i se caută anagrama. Variabila booleană `sw` este folosită tocmai pentru identificarea acestora. Dacă un cuvânt este anagramă, valoarea lui `sw` rămâne `true`, dacă însă o literă a lui `an` nu se găseşte în cuvântul care este o posibilă anagramă a acestuia, atunci `sw` devine `false`.

Scanned with OKEN Scanner

```
Program anagrama;
var a:array[1..1000] of string[10];
    n,i,j,k,poz,min,L:Integer;
    c,an:string[10];
    sw:Boolean; f:Text;

procedure citeste;
begin
  Assign(f,'INPUT.TXT'); Reset(f); Readln(f,n);
  for i:=1 to n*n do Readln(f,a[i]);
  Readln(f,an); Close(f)
end;

procedure minim;
begin
  min:=Length(a[1]); poz:=1;
  for i:=2 to n*n do
    if min>Length(a[i])
    then begin min:=Length(a[i]); poz:=i end;
  Writeln(a[poz])
end;

procedure anagrame;
begin
  L:=Length(an);
  for i:=1 to n*n do
  begin
    if Length(a[i])=L
    then
      begin
        j:=1; sw:=true;
        while sw and (j<=L) do
        begin
          sw:=false;
          for k:=1 to L do if a[i][j]=an[k] then sw:=true;
          Inc(j)
        end;
        if sw
        then Writeln('cuvantul ',a[i],' este anagrama lui ',an)
      end
  end
end;

Begin
  citeste;
  minim;
  anagrame
End.
```

Scanned with OKEN Scanner

17. Cifre romane

Să se scrie un program care transformă un număr scris cu cifre romane în scrierea cu cifre arabe.

În fişierul `INPUT.TXT` sunt scrise mai multe numere în scriere romană (corecte), fiecare pe câte o linie. Este posibil ca unele din cifrele romane să fie scrise cu litere mici. Răspunsurile se vor afişa pe ecran, fiecare număr pe câte o nouă linie.

Exemplu:

`INPUT.TXT:`	Pe ecran afişa:
`CDLIV`	`454`
`MMMMDLXXIII`	`4573`
`MCLXVI`	`1166`
`CMXLVIII`	`948`

Soluţie:

Pentru scrierea unui număr cu cifre romane se folosesc caracterele: M, D, C, L, X, V, I care corespund, în ordine, valorilor: 1000, 500, 100, 50, 10, 5, 1.

Conversia unui număr scris cu cifre romane în scriere cu cifre arabe se realizează în felul următor:

– fie `p` valoarea corespunzătoare cifrei romane curente;

– această valoare se adună la valoarea curentă (`nr`) a numărului scris cu cifre arabe ori de câte ori valoarea ei este mai mare decât valoarea `u` a cifrei romane care o succede;

– în caz contrar la numărul `nr` se adună diferenţa dintre valoarea `u` a cifrei succesoare şi valoarea `p` a cifrei curente.

```
Program conversie_numar_din_scrierea_romana_in_scrierea_araba;
const input_file='';              { numele fisierului de intrare }
                { daca este string-ul vid ('') atunci datele }
                { datele se vor citi de la intrarea standard }

function ValCif(c:Char):Integer;
                                  { intoarce valoarea cifrei romane }
begin
  case Upcase(c) of
    'M': ValCif:=1000;
    'D': ValCif:=500;
    'C': ValCif:=100;
    'L': ValCif:=50;
    'X': ValCif:=10;
    'V': ValCif:=5;
    'I': ValCif:=1
    else ValCif:=0
  end
end;
```

Scanned with OKEN Scanner

```
function ValNr(s:string):Integer;          { intoarce valoarea }
var nr,i,a,b:Integer;        { numarului scris cu cifre romane }
begin
  nr:=0;                    { a reprezinta cifra precedenta     }
  a:=ValCif(s[1]);          { b reprezinta cifra curenta        }
  for i:=1 to Length(s) do
  begin
    b:=ValCif(s[i]);
    Inc(nr,b);
    if a<b then Dec(nr,2*a);
    a:=b
  end;
  ValNr:=nr
end;

var s:string; { modulul principal }
    f:Text;

Begin
  Assign(f,input_file); Reset(f);
  while not Eof(f) do
  begin Readln(f,s); Writeln(ValNr(s)) end;
  Close(f)
End.
```

18. Scrabble

O tablă de scrabble de dimensiune nxn este completată cu n^2 litere mici ale alfabetului englez. Jack trebuie să descopere cel mai lung cuvânt cu proprietatea palindromului, ştiind că această proprietate înseamnă că citindu-l de la dreapta la stânga sau de la stânga la dreapta se obţine acelaşi cuvânt.

Cuvintele în scrabble se formează începând cu orice poziţie pe linii şi coloane. Scrieţi un program care determină palindromul de lungime maximă.

Datele de intrare se găsesc în fişierul `INPUT.TXT` care conţine n linii; fiecare linie conţine n litere între care nu există nici un separator.

Răspunsul va fi afişat pe ecran. În cazul în care există mai multe soluţii se vor afişa toate.

Exemplu:

`INPUT.TXT:`	Se va afişa pe ecran:
`acnzrr`	`cojoc`
`moazic`	
`ijdpar`	
`cokoad`	
`tclmic`	
`rpimtu`	

Scanned with OKEN Scanner

Soluţie:

Pentru rezolvarea problemei se efectuează următoarele calcule:
– se caută dimensiunea maximă `maxl` a palindroamelor pe cele `n` linii;
– se reţine în variabila `lin` indicele liniei pe care se găseşte palindromul de lungime maximă şi în variabila `pozli` indicele coloanei în care se găseşte prima literă a acestuia.
– se caută dimensiunea maximă `maxc` a palindroamelor pe cele `n` coloane;
– se reţine în variabila `col` indicele coloanei pe care se găseşte palindromul de lungime maximă şi în variabila `pozci` indicele liniei pe care se găseşte prima literă a acestuia;
– se afişează cel mai lung palindrom dintre cele două găsite.

Vectorul `b` este folosit pentru o mai uşoară verificare a proprietăţii de palindrom a şirului curent.

```
Program palindrom;
var a:array[1..100,1..100] of Char;
    b:array[1..100] of Char;
    n,i,j,l,c,k,m,p,col,lin,pozli,pozci,maxl,maxc:Integer;
    f:Text;
Begin
  Assign(f,'INPUT.TXT'); Reset(f);
  n:=0;
  while not Eof(f) do
  begin
    Inc(n); j:=0;
    while not Eoln(f) do
    begin Inc(j); Read(f,a[n,j]) end;
    Readln(f)
  end;
  Close(f); maxl:=0;
  for l:=1 to n do
  begin
    for i:=1 to n-1 do
    begin
      for j:=i+1 to n do
      begin
        p:=j-i+1;
        for k:=1 to p do b[k]:=a[l,i+k-1];
        m:=0;
        for k:=1 to Trunc(p/2) do if b[k]<>b[p-k+1] then m:=1;
        if m=0
        then
          if maxl<p then begin maxl:=p; lin:=l; pozli:=i end
      end
    end
  end;
```

```
  maxc:=0;
  for c:=1 to n do
  begin
    for i:=1 to n-1 do
    begin
      for j:=i+1 to n do
      begin
        p:=j-i+1;
        for k:=1 to p do b[k]:=a[i+k-1,c];
        m:=0;
        for k:=1 to Trunc(p/2) do
          if b[k]<>b[p-k+1] then m:=1;
        if m=0
        then
          if maxc<p
          then begin maxc:=p; col:=c; pozci:=i end
      end
    end
  end;
  if maxl>maxc
  then for i:=1 to maxl do Write(a[lin,pozli+i-1])
  else for i:=1 to maxc do Write(a[pozci+i-1,col])
End.
```

19. Propoziţii

Se consideră un text scris pe n linii (în fişierul de intrare `INPUT.TXT`). Fiecare linie constituie o propoziţie. Cuvintele sunt formate doar din litere mari şi mici şi sunt despărţite prin: spaţiu, virgulă şi punct. Nici o propoziţie nu are mai mult de 250 de caractere. Textul nu conţine cuvinte în care să apară liniuţa de despărţire (exemplu `:'intr-un'`).

Să se determine propoziţia cu cele mai multe cuvinte şi să se afişeze pe ecran scriind fiecare cuvânt pe linie nouă având prima literă majusculă; separatorii nu se vor afişa.

Exemplu:

```
INPUT.TXT:
Ce faci...
Alo, este ora sapte si treizeci de minute, fix.
buna dimineata
Unde esti
O sa reusim...
```

Se va afişa:

```
Alo
Este
Ora
Sapte
Si
Treizeci
De
Minute
Fix
```

Scanned with OKEN Scanner

Soluţie:

Rezolvarea nu necesită un algoritm complicat. Se citeşte fiecare linie din fişier într-o variabilă de tip **string**. Apoi se numără cuvintele prin ignorarea caracterelor care nu sunt litere. În variabila `max` se păstrează numărul maxim de cuvinte de pe o linie, iar în variabila `pr` propoziţia cu acest număr maxim de cuvinte.

Se afişează – respectând cerinţele – fiecare cuvânt al propoziţiei `pr` cu majusculă, pe o nouă linie.

```
Program nr_cuv;
const lit=['a'..'z']+['A'..'Z'];
var a,b,pr:string;
    i,nr,max:Integer;
    f:Text;

Begin
  Assign(f,'INPUT.TXT'); Reset(f);
  while not Eof(f) do
  begin
    Readln(f,a);
    nr:=0;
    i:=1;
    while i<=Length(a) do
    begin
      while not (a[i]in lit) do Inc(i);
      if i<=Length(a) then Inc(nr);
      while a[i] in lit do Inc(i)
    end;
    if nr>max
    then
      begin max:=nr; pr:=a end
  end;
  Close(f);
  i:=1;
  while i<=Length(pr) do
  begin
    Writeln;
    while not (pr[i] in lit) and (i<=Length(pr)) do Inc(i);
    pr[i]:=Upcase(pr[i]);
    while (pr[i] in lit) and (i<=Length(pr)) do
    begin
      Write(pr[i]);
      Inc(i)
    end
  end
End.
```

Scanned with OKEN Scanner

20. Jumătate

Se consideră o mulțime de n numere întregi.

Să se partiționeze această mulțime în două submulțimi astfel încât oricare element al primei submulțimi să fie mai mic decât b (o valoare întreagă cunoscută) iar elementele celei de a doua submulțimi să fie mai mari sau cel mult egale cu acest număr b.

Să se realizeze un program care rezolvă problema, folosind tehnica Divide et Impera.

Pe prima linie a fișierului INPUT.TXT sunt scrise numerele n și b ; pe a doua linie este scris șirul celor n numere despărțite prin spațiu.

Cele două submulțimi se vor afișa pe două linii diferite în fișierul OUTPUT.TXT.

Exemplu:

```
INPUT.TXT:              OUTPUT.TXT:
7 5                     -11 -1 1 2 3
1 3 2 -1 6 -11 9        6 9
```

Soluție:

În rezolvarea problemei se vor efectua doi pași:

– ordonarea numerelor în cadrul vectorului s;

– determinarea indicelui elementului din vectorul s care reprezintă ultimul element al primei submulțimi. În subprogramul recursiv care realizează acest al doilea pas indicele căutat este identificat prin înjumătățirea intervalului de căutare. Inițial intervalul de căutare este [1,n].

```
Program divimp;
var s:array[1..100] of Integer;
    m,k,n,b1,i,c:Integer; f,g:Text;

procedure sortare;
var i,j:Integer;
begin
  for i:=1 to n-1 do
    for j:=i+1 to n do
      if s[i]>s[j] then begin c:=s[i]; s[i]:=s[j]; s[j]:=c end
end;

procedure recurs(a,b:Integer);
begin
  if (a=b)or((b-a)=1) then k:=a
                      else begin
                             m:=Trunc((a+b)/2);
                             if s[m]<b1 then recurs(m,b)
                             else recurs(a,m)
                           end
end;
```

```
procedure citeste;
var i:Integer;
begin
  Assign(f,'INPUT.TXT'); Reset(f); Read(f,n,b1);
  for i:=1 to n do Read(f,s[i]);
  Close(f)
end;

Begin
  citeste;
  sortare;
  recurs(1,n);
  Assign(g,'OUTPUT.TXT'); Rewrite(g);
  for i:= 1 to k do Write(g,s[i],' '); Writeln(g);
  for i:= k+1 to n do Write(g,s[i],' ');
  Close(g)
End.
```

21. Polinoame

Se consideră un polinom *P(X)*, de grad `n`, reprezentat cu ajutorul unei liste simplu înlănțuite. Fiecare element al listei codifică un monom al polinomului păstrând coeficientul (număr real) şi gradul monomului (număr întreg).

Să se scrie un program care creează lista necesară reprezentării polinomului *P(X)* şi care, pentru un număr real *a*, calculează valoarea *P(a)*.

Datele de intrare se introduc de la tastatură, iar valoarea *P(a)* se va afişa pe ecran.

Exemplu:

Pentru polinomul $2X^3+X^2-2X+1$ şi `a=1.0` datele vor fi introduse astfel:

`2 3` – monomul de grad `3`
`1 2` – monomul de grad `2`
`-2 1` – monomul de grad `1`
`1 0` – monomul de grad `0`
`1.0` – valoarea reală `a`

Răspunsul va fi: `2.0`

Soluție:

Rezolvarea problemei necesită parcurgerea a două etape: crearea listei care reține polinomul şi calculul valorii *P(a)*.

Primul pas este realizat în procedura `creare`, primul monom fiind alocat însă în programul principal odată cu reținerea adresei acestuia în variabila `nod`.

Introducerea monoamelor este întreruptă când gradul monomului introdus este `0` deci s-a introdus inclusiv termenul liber. Calculul valorii polinomului *P(a)* este realizat de procedura `valoarea`, prin însumarea valorilor fiecărui monom în parte.

Scanned with OKEN Scanner

```
Program polinom;
type lista=^ inr;
     inr=record
           c,e:Integer;
           leg:lista
         end;
var p,q,nod:lista;
    val,a:Real;

procedure creare(g:Integer);
begin
  p:=nod;
  while g<>0 do
  begin
    New(q);
    with q^ do
    begin
      Write('Coeficient:'); Read(c);
      Write('Grad:'); Read(e);
      leg:=nil
    end;
    p^.leg:=q; p:=q;
    g:=q^.e
  end
end;

procedure valoare;
begin
  p:=nod; val:=0;
  while p<>nil do
  begin
    val:=val+p^.c*(Exp(p^.e*Ln(a)));
    p:=p^.leg
  end;
  Writeln(val:10:2)
end;

Begin
  New(nod);
  with nod^ do
  begin Write('Coeficient:'); Read(c);
        Write('Gradul:'); Readln(e);
        leg:=nil
  end;
  creare(nod^.e);
  Read(a);
  valoare
End.
```

Scanned with OKEN Scanner

22. Stive

Să se simuleze crearea unui vraf de farfurii (despachetate dintr-o cutie). Farfuriile sunt desenate – pe partea interioară – cu diferite modele. După crearea stivei corespunzătoare vrafului, să se caute o anumită farfurie (având un anumit model), "aşezând" totodată farfuriile puse deoparte într-un vraf nou. Să se afişeze un mesaj corespunzător în urma căutării şi numărul farfuriilor din cel de al doilea vraf.

Soluţie:

```
Program stive;
type vraf=^farfurie;
     farfurie=record
                model:string[20];
                urm:vraf
              end;
var cap,p,q,cap_nou:vraf;
    model_cautat:string[20];
    raspuns:Char;
    gasit:Boolean;
    n:Byte;
Begin
  cap:=nil;
  raspuns:='d';
  while Upcase(raspuns)<>'N' do                    { creare stiva }
  begin
    New(p);
    with p^ do
    begin Write('model: '); Readln(model); urm:=cap end;
    Write('Mai doriti sa introduceti (d/n)? ');
    Readln(raspuns);
    cap:=p
  end;
  Write('Modelul cautat: '); Readln(model_cautat);
  gasit:=false; cap_nou:=nil;
  while not gasit and (cap<>nil) do          { cautare farfurie }
  begin
    if cap^.model=model_cautat
    then gasit:=true
    else begin New(p);
               p^.model:=cap^.model;
               p^.urm:=cap_nou; cap_nou:=p;
               q:=cap; cap:=cap^.urm; Dispose(q)
         end
  end;
```

```
  p:=cap_nou;
  if gasit then Writeln('Farfuria cautata nu s-a spart!')
           else Writeln('Farfuria cautata s-a spart!');
  p:=cap_nou;
  n:=0;
  while p<>nil do              { numarare farfurii in vraful nou }
  begin
    Inc(n);
    p:=p^.urm
  end;
  Write('In vraful nou avem ',n,' farfurii.');
  Readln
End.
```

23. Cozi

Să se realizeze operaţiile de creare şi vizualizare a unei cozi implementate dinamic, precum şi eliminarea unui element din coadă.

Soluţie:

```
Program coada;
uses crt;
type reper=^el;
     el=record
          cod:Word;
          urm:reper
        end;
var ultim,prim,p:reper;
    w:Char;

procedure intrare;
begin
  if prim=nil
  then begin          { coada e vida, se adauga primul element }
         New(prim);
         Write('introduceti codul '); Readln(prim^.cod);
         prim^.urm:=nil;
         ultim:=prim
       end
  else begin New(p);
             Write('introduceti codul '); Readln(p^.cod);
             p^.urm:=nil;
             ultim^.urm:=p;     { legarea elementului la coada }
             ultim:=p       { elementul adaugat devine ultimul }
    end
end;
```

```
procedure stergere;
begin
  if prim=nil
  then Writeln('Coada este vida.')
  else
    begin
      Writeln('Iese nodul cu codul: ',prim^.cod,'.');
      p:=prim; prim:=prim^.urm; Dispose(p)
    end;
  Readkey
end;

procedure listare;
begin
  if prim=nil
  then Writeln('Coada e vida.')
  else
    begin
      p:=prim;
      Writeln('Nodurile cozii: ');
      repeat
        Write(p^.cod,'  ');
        p:=p^.urm
      until p=nil
    end;
  Readkey
end;

Begin
  Clrscr;
  prim:=nil;
  repeat
    Clrscr;
    Writeln('1. creare/adaugare ');
    Writeln('2. stergere nod ');
    Writeln('3. vizualizare coada ');
    Writeln('4. terminare ');
    Writeln('alege optiunea: ');
    Readln(w);
    case w of
      '1':intrare;
      '2':stergere;
      '3':listare
    end
  until w='4';
  Readkey
End.
```

Scanned with OKEN Scanner

24. Reprezentare în baza 16

Să se creeze o listă dublu înlănţuită având în elemente reprezentările în baza 16 a numerelor citite din fişierul standard de intrare.

Să se afişeze lista creată.

Soluţie:

Lista dublu înlănţuită creată are două santinele: `sant1` şi `sant2`. Numerele se citesc pe rând din fişierul standard de intrare în variabila `nr` de tip `real`. Funcţia `hexa` are un parametru de tip real. Transformă numărul primit ca parametru într-un şir care conţine reprezentarea sa în baza 16. Funcţia are două părţi distincte: transformarea părţii întregi şi transformarea părţii fracţionare. Partea fracţionară va avea cel mult 5 zecimale exacte.

```
Program reprezentare_in_baza_16;
const a:array[0..15] of Char=('0','1','2','3','4','5','6','7',
                              '8','9','A','B','C','D','E','F');
type reper=^element;
     element=record
       nr_hexa:string[20];
       pred,urm:reper
     end;
var nr,x:Real;
    p,sant1,sant2:reper;

function hexa(n:real):string;
var s,x:string[20];
    pi:Word; pr:Real;
    i,j:Byte;
begin
  pi:=Trunc(Int(Abs(n)));    { contine partea intreaga a lui n }
  i:=1;
  repeat
    s[i]:=a[pi mod 16];
    pi:=pi div 16;     { s contine cifrele reprezentarii partii }
    i:=i+1             { intregi a numarului n in baza 16 in    }
  until pi=0;          { ordinea inversa celei dorite           }
  i:=i-1;
  j:=1;
  if n<0 then begin x[1]:='-';j:=2 end;
  while i>0 do begin x[j]:=s[i]; i:=i-1; j:=j+1 end;
  x[j]:='.';           { x contine cifrele reprezentarii partii }
  j:=j+1;              { intregi a numarului n in baza 16 in    }
                       { ordinea dorita                         }
  pr:=Abs(n)-Int(Abs(n));
                { pr contine partea fractionara a numarului n }
```

```
  i:=1;
  while (i<6) and (Frac(pr)<>0) do
  begin
    x[j]:=a[Trunc(Int(pr*16))]; j:=j+1;
    pr:=pr*16; pr:=pr-Int(pr);
    i:=i+1
  end;
  x[0]:=Chr(j-1);              { se modifica lungimea sirului x }
  hexa:=x
end;

Begin
  New(sant1); New(sant2);
  sant1^.urm:=sant2; sant2^.pred:=sant1;
  Write('Numarul:');
  while not Eof do
  begin Readln(nr); p:=sant2; p^.nr_hexa:=hexa(nr);
        New(sant2); sant2^.pred:=p; p^.urm:=sant2;
        Write('Numarul:')
  end;
  p:=sant1^.urm;
  if p=sant2 then Writeln('Lista vida')
             else
               while p<>sant2 do
               begin Writeln(p^.nr_hexa); p:=p^.urm end
End.
```

25. Ştergeri din listă

Să se creeze o listă înlănţuită formată din numere întregi, distincte, citite din fişierul standard de intrare (numerele din fişier nu sunt neapărat distincte).

Să se determine cele mai mici trei elemente ale listei, apoi să se şteargă aceste elemente din listă. Să se afişeze lista înainte şi după ştergere.

Soluţie:

În implementarea listei simplu înlănţuite se folosesc două santinele: `sant1` şi `sant2`. `Sant1` e necesară pentru ca adăugarea primului element să nu se facă diferit faţă de a celorlalte, iar `sant2` se foloseşte pentru evitarea testului de determinare dacă lista e vidă.

```
Program stergeri_din_lista;
type reper=^element;
    element=record
              nr:Integer;
              urm:reper
            end;
```

```
var nr:Integer;
    exista:Boolean;
    i:Byte;
    p,sant1,sant2:reper;

function gaseste_min:reper;
{ functia va returna adresa elementului care contine intregul }
{ cel mai mic din lista sau nil daca lista este vida          }
var p,q:reper;
    min:Integer;
begin
  q:=nil;
  if sant1^.urm<>sant2
  then
    begin
      p:=sant1^.urm;
      min:=p^.nr;
      q:=p; p:=p^.urm;
      while p<>sant2 do
      begin
        if p^.nr<min
        then begin min:=p^.nr; q:=p end;
        p:=p^.urm
      end
    end;
  gaseste_min:=q
end;

procedure sterge(p:reper);
{ Sterge din lista elementul referit de p. Elementul urmator  }
{ celui care se sterge se muta mai intai peste acesta, iar    }
{ apoi se elibereaza zona ocupata de elementul urmator. Acest }
{ artificiu este necesar deoarece lista este simplu inlantuita}
{ si nu se cunoaste adresa elementului precedent lui p. Daca  }
{ succesorul lui p este sant2, atunci se modifica si valoarea }
{ acestei variabile. }
var q:reper;
begin
  q:=p^.urm;
  p^:=q^;
  Dispose(q);
  if p^.urm=nil then sant2:=p
end;

Begin
  New(sant1); New(sant2);
  sant1^.urm:=sant2; sant2^.urm:=nil;
  Write('Numarul:');
```

Scanned with OKEN Scanner

```
  while not Eof do
  begin
    Readln(nr);
    p:=sant1^.urm;
    exista:=false;
    while (p<>sant2) and (not exista) do
    begin
      if p^.nr=nr then exista:=true;
      p:=p^.urm
    end;
    if not exista
    then
      begin
        p:=sant2;
        p^.nr:=nr;
        new(sant2);
        sant2^.urm:=nil;
        p^.urm:=sant2
      end;
    Write('Numarul:')
  end;
  p:=sant1^.urm;
  Writeln('Numerele distincte inainte de stergere:');
  if p=sant2
  then Writeln('Lista vida')
  else
    begin
      while p<>sant2 do
      begin
        Writeln(p^.nr); p:=p^.urm
      end;
      for i:=1 to 3 do
        if sant1^.urm<>sant2
        then sterge(gaseste_min);
      p:=sant1^.urm;
      Writeln('Lista dupa stergere este:');
      if p=sant2
      then Writeln(' lista vida')
      else
        while p<>sant2 do
        begin
          Writeln(p^.nr);
          p:=p^.urm
        end
    end
End.
```

Scanned with OKEN Scanner

26. Joc

Să se simuleze, utilizând alocare dinamică, următorul joc: n copii sunt în cerc. Copiii efectuează o numărătoare şi fiecare al k-lea copil iese din cerc. Câştigă jocul acel copil care rămâne ultimul în cerc. Numerele n şi k se citesc de la tastatură. Numele copiilor se citesc de pe câte o linie din fişierul text JOC.TXT.

Se cere să se afişeze numele copiilor în ordinea ieşirii lor din joc.

Soluţie:

Aşezarea copiilor în cerc va fi simulată printr-o listă circulară care se realizează prin legarea ultimului element din listă de primul. Prin codificarea în acest fel a jocului, ieşirea unui copil din cerc înseamnă ştergerea elementului corespunzător din listă.

```
Program joc;
type reper=^element;
     element=record
                nume:string[20];
                urm:reper
             end;
var f:Text;
    k,i,n:Byte;
    p,q,sant,ultim:reper;
Begin
  Assign(f,'JOC.TXT'); Reset(f);
  New(sant); sant^.urm:=nil; ultim:=sant;
  Write('n='); Readln(n);
  for i:=1 to n do
  begin
    New(p); Readln(f,p^.nume); p^.urm:=nil;
    ultim^.urm:=p; ultim:=p
  end;
  Close(f);
  Write('k='); Readln(k);
  ultim^.urm:=sant^.urm;  { dupa ultimul copil urmeaza primul }
  p:=sant^.urm;           { numaratoarea incepe de la primul copil }
  while p<>p^.urm do
  begin
    for i:=1 to k-1 do p:=p^.urm;
    Writeln('Iese din cerc ',p^.nume);
    q:=p^.urm;
    p^:=q^;
    Dispose(q)
  end;
  Writeln('A castigat ',p^.nume);
End.
```

Scanned with OKEN Scanner

27. Cuvinte distincte

Se citesc cuvinte din fişierul text CUV.TXT în care pe o linie pot fi mai multe cuvinte separate prin spaţii.

Să se creeze o listă înlănţuită, ordonată alfabetic care să cuprindă cuvintele distincte din fişier.

Să se afişeze lista creată.

Soluţie:

```
Program cuvinte_ordonate_alfabetic;
type reper=^element;
     element=record
               cuvant:string[20];
               urm:reper
             end;
var f:Text;
    cuv:string[20];
    linie:string;
    p,q,sant1,sant2:reper;
    i:Integer;

Begin
  New(sant1);
  New(sant2);
  sant1^.urm:=sant2;
  sant2^.cuvant:='zzzzzzzzzzzzzzzzzzzz';
      { pentru ca si primul cuvant din lista sa fie introdus  }
      { inaintea unui cuvant care-i urmeaza alfabetic, campul }
      { cuvant din sant2 se initializeaza cu 'zzz...zz'       }
  Assign(f,'CUV.TXT');
  Reset(f);
  while not Eof(f) do
  begin
    Readln(f,linie);
    i:=1;
    while i<=Length(linie) do
    begin
      cuv:='';
      while (i<=Length(linie)) and (linie[i]<>' ') do
      begin
        cuv:=cuv+linie[i];
        i:=i+1
      end;
      while (i<=Length(linie)) and (linie[i]=' ') do i:=i+1;
```

Scanned with OKEN Scanner

```
      if cuv<>''
      then
        begin
          p:=sant1^.urm;
          while p^.cuvant<cuv do p:=p^.urm;
          if p^.cuvant<>cuv { daca cuvantul cuv nu mai exista }
          then              { in lista va fi introdus in fata }
            begin           { celui care-i urmeaza alfabetic  }
              New(q);       { adaugarea elementului in fata   }
              q^:=p^;       { lui p^                          }
              p^.cuvant:=cuv;
              p^.urm:=q;
              if p=sant2 then sant2:=q
            end
        end
    end
  end;
  Close(f);
  p:=sant1^.urm;
  while p<>sant2 do
  begin
    Writeln(p^.cuvant);
    p:=p^.urm
  end
End.
```

28. Arbore

Să se citească (din fişierul standard de intrare) un număr necunoscut de cuvinte (nu neapărat distincte) separate prin caractere albe (spaţiu, Enter). Să se creeze un arbore binar de căutare care să cuprindă numai cuvintele distincte.

Să se afişeze cuvintele din arbore, parcurs în preordine (cu indentare).

Soluţie:

```
Program arbore;
type cuvant=string[20];
     arb=^nod;
     nod=record
           cuv:cuvant;
           st,dr:arb
         end;
var cap:arb;                              { radacina arborelui }
    f:Text;
    s,s1:cuvant;
    i:Byte;
```

```
function adaug(s:cuvant; p:arb):arb;
 { adauga in arbore doar cand cuvantul nu a mai fost introdus }
 { functie recursiva care returneaza pointer spre nodul creat }
begin
  if p =nil
  then begin new(p); p^.cuv:=s; p^.st:=nil; p^.dr:=nil end
  else
    if p^.cuv<s
    then p^.dr:=adaug(s,p^.dr)
    else if p^.cuv > s then p^.st:=adaug(s,p^.st);
  adaug:=p
end;

procedure afisare(c:arb; i:Integer);
 { procedura afiseaza arborele in preordine, arbore nenul      }
 { i asigura indentarea -numarul de spatii din fata numarului }
begin
  Write(' ':i); Writeln(c^.cuv);
  if c^.st<>nil then afisare(c^.st,i+1);
  if c^.dr<>nil then afisare(c^.dr,i+1)
end;
Begin
  Assign(f,'CUV.TXT'); Reset(f);
  cap:=nil;
  while not Eof(f) do
  begin
    Readln(f,s); { se elimina spatiile de la inceputul liniei }
    while s[1]=' ' do delete(s,1,1);
    if s<>''
    then                                  { daca linia nu e vida }
      repeat                              { se extrag cuvintele  }
        i:=Pos(' ',s);
        if i>0 then begin
                      s1:=copy(s,1,i-1);
                      cap:=adaug(s1,cap);
                      Delete(s,1,i);
                        { cuvantul extras se elimina din linie }
                      while s[1]=' ' do Delete(s,1,1)
                    end
               else cap:=adaug(s,cap)
      until i=0
  end;
  if cap<>nil then afisare(cap,1)
              else Writeln('Arbore vid');
  Readln; Close(f)
End.
```

Scanned with OKEN Scanner

29. Săritură cu schiurile

Să se construiască o listă alocată dinamic cu articolele care conţin numele şi punctajul obţinut de concurenţi la săritura cu schiurile.

Să se afişeze lista concurenţilor descrescător după punctajul realizat. În caz de egalitate de puncte, concurenţii trebuie ordonaţi alfabetic.

Citirea numelor şi a punctajelor se face din fişierul standard de intrare.

Soluţie:

```
Program schiori;
      { se creeaza o lista ordonata descrescator dupa punctaj }
      { si in caz de egalitate de puncte concurentii apar in  }
      { ordine alfabetica                                     }
type nume=string[20];
     list=^nod;
     nod=record
       num:nume;
       punct:real;
       urm:list
     end;
var cap,s:list;                          { cap - inceputul listei }
    i:Byte; f:Text;
procedure adaug(s:list);
{ Adauga nodul s in lista ordonata dupa punctaj care are      }
{ santinela cap. Daca sunt 2 schiori cu acelasi punctaj       }
{ se adauga in ordine alfabetica.                             }
var p:list;
begin
  p:=cap;                      { caut locul in functie de punctaj }
  while (p^.urm<>nil) and (p^.urm^.punct>s^.punct) do p:=p^.urm;
  if p^.urm<>nil
  then                              { caut locul in functie de nume }
    while (p^.urm^.punct=s^.punct) and (p^.urm^.num<s^.num) do
      p:=p^.urm;
  s^.urm:=p^.urm;                          { legatura propriu-zisa }
  p^.urm:=s
end;
procedure afisare(c:list);     { afiseaza lista cu santinela c }
begin
  Writeln('Numele':20, 'Punctaj':10); c:=c^.urm;
  while c<>nil do
  begin
    Writeln(c^.num:20, c^.punct:10:2); c:=c^.urm
  end
end;
```

Scanned with OKEN Scanner

```
Begin
  New(cap);                                    { creeaza santinela }
  cap^.urm :=nil;
  cap^.num:='';
  cap^.punct:=0;
  i:=1;
  while not Eof do
  begin
    New(s);                              { alocare spatiu pentru schior }
    Readln(s^.num);
    Readln(s^.punct);
    adaug(s);                                   { adaugare in lista }
    Inc(i)
  end;
  afisare(cap);
  Readln
End.
```

30. Interclasare

Se citesc din fişierul text SIRURI.TXT două linii care conţin două şiruri neordonate de numere întregi.

Să se creeze din fiecare şir câte o listă de numere ordonate crescător. Să se construiască o a treia listă care se obţine prin interclasarea celor două şi să se afişeze toate cele trei liste.

Soluţie:

```
Program interclasare;
type list=^nod;
     nod=record
           nr:Integer;
           urm:list
     end;
var cap1,cap2,cap3:list;
    n:Integer;
    f:Text;

procedure inter(cap1, cap2:list; var cap3:list);
          { interclasarea propriu-zisa                                  }
          { cap1,cap2 :santinelele listelor initiale                    }
          { cap3: santinela listei rezultate din inteclasare }
var p,q:list;
begin
  New(cap3);
  cap3^.urm:=nil;
  cap3^.nr:=-maxint;     { crearea santinelei listei rezultate }
```

Scanned with OKEN Scanner

```
    cap1:=cap1^.urm;
    cap2:=cap2^.urm;
    p:=cap3;
    while (cap1<>nil) and (cap2<>nil) do
    begin
      New(p^.urm);
      p:=p^.urm; p^.urm:=nil;
                          { adaugarea unui nou nod in lista 3   }
      if cap1^.nr<cap2^.nr
      then begin p^.nr:=cap1^.nr; cap1:=cap1^.urm end
      else begin p^.nr:=cap2^.nr; cap2:=cap2^.urm end
    end;
    while cap1<>nil do
    begin
      New(p^.urm);
      p:=p^.urm; p^.urm:=nil;
      p^.nr:=cap1^.nr;
      cap1:=cap1^.urm
    end;
    while (cap2<>nil) do
    begin
      New(p^.urm);
      p:=p^.urm; p^.urm:=nil;
      p^.nr:=cap2^.nr;
      cap2:=cap2^.urm
    end
end;

procedure adaug(n:Integer; cap:list);
                    { adauga un element, pastrand lista ordonata }
var p,q:list;
begin
  p:=cap;           { se cauta primul nod cu informatie mai      }
                    { mare sau egala, sau sfirsitul listei       }
  while (p^.urm<>nil) and (p^.urm^.nr<n) do p:=p^.urm;
  New(q);
  q^.nr:=n;
  q^.urm:=p^.urm; p^.urm:=q
end;

procedure afisare(c:list);   { afiseaza o lista cu santinela c }
begin
  c:=c^.urm;
  while c<>nil do
  begin Write(c^.nr:4); c:=c^.urm end;
  Writeln
end;
```

Scanned with OKEN Scanner

```
Begin
  Assign(f,'SIRURI.TXT'); Reset(f) ;
  New(cap1);
  cap1^.urm :=nil; cap1^.nr:=-maxint;        { creare santinele }
  New(cap2);
  cap2^.urm :=nil; cap2^.nr:=-maxint;
  while not Eoln(f) do              { citire si adaugare in liste }
  begin Read(f,n); adaug(n,cap1) end;
  Readln(f);
  while not Eoln(f) do
  begin Read(f,n); adaug(n,cap2) end;
  Writeln('Lista 1:'); afisare(cap1);
  Writeln('Lista 2:'); afisare(cap2);
  Writeln('Lista rezultata din interclasare:');
  inter(cap1,cap2,cap3);
  afisare(cap3);
  Close(f)
End.
```

31. Frecvenţe

Se citesc două linii din fişierul text FRECV.TXT care cuprinde numere întregi.

Să se creeze o listă alocată dinamic care cuprinde numerele distincte din prima linie şi o altă listă alocată dinamic care cuprinde numerele din linia a doua.

Să se afişeze elementele primei liste şi numărul de apariţii ale fiecărui element afişat în a doua listă.

Soluţie:

```
Program frecvente;
type lista=^nod;
     nod=record
       info:Integer;
       leg:lista
     end;

var f:Text;
    l1,l2,l,p1,p2,p:lista;
    n:Integer;

procedure afisare(l:lista);        { afiseaza lista care incepe }
var p:lista;                                  { cu santinela l }
begin
  p:=l^.leg;
  while p<>nil do
  begin Write(p^.info:6); p:=p^.leg end
end;
```

```
procedure frecv(l1,l2:lista);
            { Afiseaza elementele din prima lista impreuna cu }
            { frecventele de aparitie in cea de-a doua lista; }
            { l1, l2 santinelele celor 2 liste                }
var p1,p2:lista;
    f:Byte;
begin
  Writeln('Numar':10,'Frecventa':10);        { capul de tabel }
  p1:=l1^.leg;
  while p1<>nil do
  begin                                  { se parcurge prima lista }
    p2:=l2^.leg;
    f:=0;
    while p2<>nil do
    begin       { se verifica daca elementul din prima lista  }
                { este si in a doua                           }
      if p1^.info =p2^.info then f:=f+1;
      p2:=p2^.leg
    end;
    Writeln(p1^.info:10,f:10);
    p1:=p1^.leg
  end
end;

Begin
  Assign(f,'RECV.TXT'); Reset(f);
  New(l1); New(l2);              { crearea santinelelor          }
  l1^.leg:=nil; l2^.leg:=nil; { listele au santinele initiale }
                              { citirea si crearea listelor   }
  p1:=l1;                 { prima lista - prima linie din fisier }
  while not Eoln(f) do
  begin
    Read(f,n);{ verificam daca numarul nu a mai fost introdus }
    p:=l1^.leg;
    while (p<>nil) and (p^.info <>n) do p:=p^.leg;
    if p=nil
    then
      begin
        New(p1^.leg);
        p1:=p1^.leg;
        p1^.info:=n;
        p1^.leg:=nil
      end
  end;
  Readln(f);            { a doua lista - a doua linie din fisier }
  p2:=l2;               { fara verificarea unicitatii            }
```

Scanned with OKEN Scanner

```
  while not Eoln(f) do
  begin
    New(p2^.leg); p2:=p2^.leg;
    Read(f,p2^.info);
    p2^.leg:=nil
  end;
  Writeln; Writeln('Prima lista:');
  afisare(l1);
  Writeln; Writeln('A doua lista:');
  afisare(l2); Writeln;
  frecv(l1,l2);
  Close(f)
End.
```

32. Intersecţie

Se citesc două linii din fişierul text INTER.TXT care cuprinde numere întregi.

Să se creeze din fiecare linie câte o listă alocată dinamic, formată din numerele distincte. Să se creeze şi să se afişeze o nouă listă formată din intersecţia celor două liste.

Soluţie:

```
Program inter;
type lista=^nod;
     nod=record
       info:Integer;
       leg:lista
     end;
var f:Text;
    l1,l2,l:lista;
    nr:Integer;

procedure afisare(l:lista);
                  { afiseaza lista care incepe cu santinela l }
var p:lista;
begin
  p:=l^.leg;
  while p<>nil do
  begin Write(p^.info:6); p:=p^.leg end
end;

procedure intersectie(l1,l2:lista; var l:lista);
 { creeaza lista l, formata din intersectia listelor l1 si l2 }
var p1,p2,p:lista;
begin
  New(l); l^.leg:=nil;
```

Scanned with OKEN Scanner

```
  p1:=l1^.leg; p:=l;
  while p1<>nil do
  begin                                   { se parcurge prima lista }
    p2:=l2^.leg;
    while p2<>nil do              { se verifica daca elementul din }
    begin                         { prima lista este si in a doua  }
      if p1^.info =p2^.info
      then
        begin
          new(p^.leg);    { se adauga un nou nod in lista noua }
          p:=p^.leg; p^.leg:=nil;
          p^.info:=p1^.info; p2:=nil
        end
      else p2:=p2^.leg
    end;
    p1:=p1^.leg
  end
end;

procedure adaug(l:lista; n:Integer);
var p:lista;
begin
  p:=l;                { verific daca numarul este deja in lista }
  while (p^.leg<>nil) and (p^.info<>n) do p:=p^.leg;
  if p^.info<>n
  then
    begin new(p^.leg); p:=p^.leg; p^.info:=n; p^.leg:=nil end
end;

Begin
  Assign(f,'INTER.TXT'); Reset(f);
  New(l1); New(l2);                       { crearea santinelelor }
  l1^.leg:=nil; l2^.leg:=nil; { listele au santinele initiale }
  l1^.info:=maxint;     { presupun ca numerele sunt strict mai }
  l2^.info:=maxint;     { mici decat maxint                    }
                        { citirea si crearea listelor          }
                        { prima lista - prima linie din fisier }
  while not Eoln(f) do begin Read(f,nr); adaug(l1,nr) end;
  Readln(f);          { a doua lista - a doua linie din fisier }
  while not Eoln(f) do
  begin Read(f,nr); adaug(l2,nr) end;
  Writeln; Writeln('Prima lista:'); afisare(l1);
  Writeln; Writeln(' A doua lista:'); afisare(l2);
  intersectie(l1,l2,l);
  Writeln; Writeln('Intersectia:'); afisare(l);
  Close(f)
End.
```

Scanned with OKEN Scanner

33. Matrice

Se dă o matrice A cu m linii şi n coloane. Să se scrie un program care prin intermediul unui meniu realizează următoarele operaţii:

a) elimină din matrice linia L şi coloana k fără a folosi o nouă matrice.

b) ordonează crescător elementele de pe o coloană oarecare j prin interschimbări de linii.

Datele de intrare se vor citi dintr-un fişier text.

Soluţie:

```
Program matrice;
uses Crt;
type mat=array[1..10,1..10] of Integer;
var a,b:mat;
    i,j,k,L:Byte;
    m,n:Integer;
    f:Text;

procedure afisare(a:mat);
begin
  for i:=1 to m do
  begin
    for j:=1 to n do Write(a[i,j]:4); Writeln
  end
end;

procedure eliminare;
begin
  Writeln('Matricea initiala:'); afisare(a);
  b:=a;
  repeat
    Write('Linia:'); Readln(L);
    if (L<0) or (L>m) then Writeln('Nr. de linie nu este bun.')
  until (L>0) and (L<=m);
  repeat
    Write('Coloana:'); Readln(k);
    if (k<0)or (k>n) then Writeln('Nr. de coloana nu este bun')
  until (k>0) and (k<=n);
  for i:=L to m-1 do
    for j:=1 to n do b[i,j]:=b[i+1,j];
  m:=m-1;
  for i:=1 to m do
    for j:=k to n-1 do b[i,j]:=b[i,j+1];
  n:=n-1;
  Writeln('Matricea obtinuta:'); afisare(b); Readkey;
  m:=m+1; n:=n+1
end;
```

Scanned with OKEN Scanner

```
procedure ordonare;
var aux:Integer;
    sw:Boolean;
begin
  Writeln('Matricea initiala:');
  afisare(a);
  b:=a;
  repeat
    Write('Coloana:'); Readln(k);
    if (k<0)or (k>n) then Writeln('Nr. de coloana nu este bun')
  until (k>0) and (k<=n);
  repeat
    sw:=true;
    for i:=1 to m-1 do
      if b[i,k]>b[i+1,k]
      then
        begin
          for j:=1 to n do
          begin
            aux:=b[i,j];
            b[i,j]:=b[i+1,j];
            b[i+1,j]:=aux
          end;
          sw:=false
        end
  until sw;
  Writeln('Matricea obtinuta:'); afisare(b); Readkey
end;

Begin
  Assign(f,'OP_MAT.INT'); Reset(f);
  Read(f,m); Readln(f,n);
  for i:=1 to m do
  begin for j:=1 to n do Read(f,a[i,j]); Readln(f) end;
  repeat
    Writeln('1 eliminarea unei linii si a unei coloane');
    Writeln('2 ordonarea el. de pe o coloana in ordine'+
                    ' crescatoare prin interschimbari de linii');
    Writeln('3 exit');
    case Readkey of
      '1':eliminare;
      '2':ordonare
    end
  until Readkey='3';
  Close(f)
End.
```

Scanned with OKEN Scanner

34. Numere complexe

Să se citească dintr-un fişier text un şir (z_n) de numere complexe. Să se scrie un program care realizează următoarele operaţii:
a) determină şi afişează şirul modulelor numerelor complexe, ordonat crescător, utilizând un şir de indici;
b) afişează forma trigonometrică (modulul şi argumentul) pentru fiecare element al şirului de numere complexe, precizând cadranul în care se găseşte argumentul.

Soluţie:

```
Program nr_complexe;
type complex=record
                re,im,modul,argument:Real;
                cadran:Byte
              end;
     mat=array[1..10] of complex;
var z:mat;
    n,i:Byte;       { n-dimensiunea sirului numerelor complexe }
    maxim:Real;
    f:Text; cale:string;

procedure citire;
begin
  Write('Dati numele fisierului:'); Readln(cale);
  Assign(f,cale); Reset(f); Readln(f,n);
  for i:=1 to n do
  begin Readln(f,z[i].re,z[i].im);
        z[i].modul:=Sqrt(Sqr(z[i].re)+Sqr(z[i].im))
  end;
  Close(f)
end;

procedure ordonare;
var sw:Boolean; aux:complex;
begin
  repeat
    sw:=true;
    for i:=1 to n-1 do
      if z[i].modul>z[i+1].modul
      then begin aux:=z[i]; z[i]:=z[i+1]; z[i+1]:=aux;
                 sw:=false
           end
  until sw;
  Writeln('Sirul modulelor:');
  for i:=1 to n do Write(z[i].modul:5:2,' '); Writeln
end;
```

```
procedure cadranul;
begin
  if z[i].im>=0
  then
    if z[i].re>=0 then z[i].cadran:=1
                  else z[i].cadran:=2
  else
    if z[i].re<0 then z[i].cadran:=3
                 else z[i].cadran:=4
end;

procedure trig;
begin
  for i:=1 to n do
  begin
    if z[i].re<>0
    then
      begin
        z[i].argument:=Arctan(z[i].im/z[i].re)*(180/PI);
        cadranul
      end
    else Writeln('Impartire imposibila!')
  end
end;

procedure scrie;
begin
  for i:=1 to n do
  begin
    Writeln;
    Writeln(z[i].re:5:2,'+',z[i].im:5:2,'*i');
    Write('modulul:',z[i].modul:5:2);
    Writeln('argument:',z[i].argument:5:2,'cadran:',z[i].cadran)
  end
end;

Begin
  citire;
  ordonare;
  trig;
  scrie;
  Readln
End.
```

Scanned with OKEN Scanner

35. Polinoame

Se consideră două polinoame de forma:

$P(X)=a_0X^n+a_1X^{n-1}+...+a_n$ şi

$Q(X)=b_0X^m+b_1X^{m-1}+...+b_m$.

Să se scrie un program care realizează următoarele operaţii cu polinoame:

a) adunarea şi scăderea a două polinoame;

b) înmulţirea a două polinoame;

c) calculul derivatei unui polinom

d) evaluarea unui polinom într-un punct dat.

Datele de intrare se vor citi dintr-un fişier text:

– pe prima linie este scris gradul primului polinom;

– pe linia a doua sunt scrişi coeficienţii primului polinom, delimitaţi de unul sau mai multe spaţii, primul fiind termenul liber;

– pe linia a treia este scris graduł celui de-al doilea polinom;

– pe linia următoare sunt scrişi coeficienţii celui de-al doilea polinom în mod similar cu primul polinom.

Soluţie:

```
Program polinoame;
uses Crt;
type pol=array[0..100] of Real;
var a,b,rezult:pol;
    na,nb,nz,i,j:Byte;
    opt:Char;
    fi:Text;
    fis:string;

procedure cit(var x:pol; var dim: Byte);
begin
  Readln(fi,dim);
  for i:=0 to dim do Read(fi,x[i]);
end;

procedure citire;
begin
  Writeln('numele fisierului de intrare');
  Readln(fis);
  Assign(fi,fis);
  Reset(fi);
  cit(a,na);
  cit(b,nb)
end;
```

Scanned with OKEN Scanner

```
procedure afisare(x:pol; dim:Byte);
begin
  Write('  ');
  for i:=0 to dim do
    if x[i]<>0
    then
      begin
        if x[i]<0 then Write(#8#8,'- ');
        if ((x[i]<>1) and (x[i]<>-1)) or (i=0)
        then Write(abs(x[i]):3:2,'*');
        if i=0 then Write(#8,' + ')
               else
                 begin
                   Write('x');
                    if i>1 then Write('^',i,' + ')
                           else Write (' + ')
                 end
      end;
  Writeln(#8#8,' ')
end;

procedure suma;
begin
  Writeln('Suma polinoamelor:');
  afisare(a,na); afisare(b,nb); Writeln(' este:');
  if na>nb
  then
    begin
      for i:=0 to nb do rezult[i]:=a[i]+b[i];
      for i:=nb+1 to na do rezult[i]:=a[i];
      afisare(rezult,na)
    end
  else
    begin
      for i:=0 to na do rezult[i]:=a[i]+b[i];
      for i:=na+1 to nb do rezult[i]:=b[i];
      afisare(rezult,nb)
  end;
  Readkey
end;

procedure diferenta;
begin
  Writeln('diferenta polinoamelor:');
  afisare(a,na); afisare(b,nb);
  Writeln(' este:');
```

Scanned with OKEN Scanner

```
  if na>nb
  then
    begin
      for i:=0 to nb do rezult[i]:=a[i]-b[i];
      for i:=nb+1 to na do rezult[i]:=a[i];
      afisare(rezult,na)
    end
  else
    begin
      for i:=0 to na do rezult[i]:=a[i]-b[i];
      for i:=na+1 to nb do rezult[i]:=-b[i];
      afisare(rezult,nb)
    end;
  Readkey
end;

procedure produs;
begin
  Write('produsul polinoamelor:');
  afisare(a,na); afisare(b,nb); Writeln('este:');
  for i:=0 to na+nb do rezult[i]:=0;
  for i:=0 to na do
    for j:=0 to nb do rezult[i+j]:=rezult[i+j]+a[i]*b[j];
  afisare(rezult,na+nb);
  Readkey
end;

procedure derivata(x:pol;nx:Byte);
begin
  Clrscr;
  Writeln('Derivata polinomului: '); afisare(x,nx);
  Writeln('este:');
  for i:=1 to nx do rezult[i-1]:=x[i]*i;
  afisare(rezult,nx-1); Readkey
end;

procedure deriv;
var n:Char;
begin
  repeat
    Write('pentru care dintre polinoame doriti sa calculati');
    Writeln(' derivata (1,2)?');
    n:=Readkey;
    if n='1' then derivata(a,na)
             else if n='2' then derivata(b,nb)
  until n in ['1','2']
end;
```

Scanned with OKEN Scanner

```
procedure ev(x:pol; nx:Byte);
var pct,putere,valoare:Real;
begin
  Write('Punctul dat? '); Readln(pct);
  putere:=1; valoare:=0;
  for i:=0 to nx do
  begin
    valoare:=valoare+x[i]*putere;
    putere:=putere*pct
  end;
  Writeln('Valoarea polinomului:'); afisare(x,nx);
  Writeln(' In punctul ',pct,' este ',valoare); Readkey
end;

procedure eval;
var n:Char;
begin
  repeat
    Writeln('Care dintre polinoame doriti sa-l evaluati (1,2)?');
    n:=Readkey;
    if n='1' then ev(a,na)
             else if n='2' then ev(b,nb)
  until n in ['1','2']
end;

Begin
  repeat
    Clrscr;
    Writeln('1. citire');
    Writeln('2. suma polinoamelor');
    Writeln('3. diferenta polinoamelor');
    Writeln('4. inmultirea polinoamelor');
    Writeln('5. derivarea unui polinom');
    Writeln('6. evaluarea unui polinom intr-un punct');
    Writeln('7. exit'); opt:=Readkey;
    Clrscr;
    case opt of
      '1': citire;
      '2': suma;
      '3': diferenta;
      '4': produs;
      '5': deriv;
      '6': eval;
      '7': ;
      else Write('Optiunea trebuie sa fie in intervalul 1-8')
    end
  until opt='7'
End.
```

Scanned with OKEN Scanner

36. Ordonare

Se consideră un tablou bidimensional pătratic de dimensiune n×n de numere întregi.

Se cere să se inverseze liniile şi coloanele acestuia astfel încât tabloul obţinut să conţină elementele de pe diagonala principală în ordine crescătoare. Datele de intrare se citesc din fişierul de intrare INPUT.TXT sub forma următoare:

Pe prima linie este scris numărul n, reprezentând numărul de linii şi coloane.

Pe următoarele n linii se găsesc elementele tabloului linie după linie.

Rezultatul va fi scris în fişierul OUTPUT.TXT. Pe fiecare linie a fişierului se va scrie o linie a tabloului obţinut.

Exemplu:

```
INPUT.TXT:          OUTPUT.TXT:
3                   -1 8 7
2 4 3               2 3 4
1 5 6               1 6 5
-1 7 8
```

Soluţie (1):

Pentru rezolvarea problemei se determină de n-1 ori minimul în tablou, pentru linia i şi coloana i începând cu elementul a[i,i] (i=1,...,n-1).

Se reţine poziţia acestui element: a[L,c]=min.

Se inversează linia L cu linia i şi coloana c cu coloana i doar dacă L<>i sau c<>i.

```
Program diagonala;
var c,min,i,j,k,L,n,v:Integer;
    a:array[1..100,1..100] of Integer;
    f,g:Text;
Begin
  Assign(f,'INPUT.TXT'); Reset(f);
  Readln(f,n);
  for i:=1 to n do
  begin
    for j:=1 to n do Read(f,a[i,j]);
    Readln(f)
  end;
  Close(f);
  for i:=1 to n do
  begin
    min:=a[i,i];
    L:=i;
    c:=i;
    for j:=i to n do
      for k:=i to n do
        if min>a[j,k]
        then begin min:=a[j,k]; L:=j; c:=k end;
```

Scanned with OKEN Scanner

```
    if L<>i
    then
      for j:=1 to n do
      begin
        v:=a[i,j]; a[i,j]:=a[L,j]; a[L,j]:=v
      end;
    if c<>i
    then
      for j:=1 to n do
      begin
        v:=a[j,c]; a[j,c]:=a[j,i]; a[j,i]:=v
      end
  end;
  Assign(g,'OUTPUT.TXT'); Rewrite(g);
  for i:=1 to n do
  begin
    for j:=1 to n do Write(g,a[i,j],' ');
    Writeln(g)
  end;
  Close(g)
End.
```

Soluție (2):

Tabloul care trebuie ordonat este a_{ii}, $\overline{i=1,n}$. Singura complicație este că nu putem interschimba numai elementele a_{ii} şi $a_{i+1,i+1}$, ci trebuie să interschimbăm linia `i` cu linia `i+1` şi coloana `i` cu coloana `i+1`.

Variabila `a` s-a declarat ca tablou cu elemente de tip `linie`, deoarece trebuia ca liniile `a[i]` şi `a[i+1]` să aibă acelaşi tip cu `linie_aux`, pentru ca atribuirile să fie corecte.

```
Program Ordonarea_elementelor_de_pe_diagonala_principala;
type linie=array[1..10] of Integer;
var a:array[1..10] of linie;
    n,i,j:1..10;
    ordonat:Boolean;
    linie_aux:linie;
    aux:Integer;
Begin
  Write('Introduceti n='); Read(n);
  Writeln('Introduceti matricea a :');
  for i:=1 to n do                          { citirea matricei A }
  begin
    Write(' Linia ',i,' : ');
    for j:=1 to n do Read(a[i,j])
  end;
```

Scanned with OKEN Scanner

```
  repeat
    ordonat:=true;
    for i:=1 to n-1 do
    if a[i,i]>a[i+1,i+1]
    then
    begin
      linie_aux:=a[i];                    {    interschimbarea }
      a[i]:=a[i+1];                       {  liniilor i si i+1 }
      a[i+1]:=linie_aux;
      for j:=1 to n do
      begin                               {    interschimbarea }
        aux:=a[j,i];                      {coloanelor i si i+1 }
        a[j,i]:=a[j,i+1];
        a[j,i+1]:=aux
      end;
      ordonat:=false
    end
  until ordonat;
  Write('Matricea A cu diagonala principala');
  Writeln(' ordonata crescator :');
  for i:=1 to n do                              { scrierea matricei A }
  begin for j:=1 to n do Write(a[i,j]:3); Writeln end
End.
```

37. Frecvenţe (2)

Se consideră un fişier `INPUT.TXT` care conţine un text. Se cere determinarea literei cu frecvenţa de apariţie maximă. În cazul când există mai multe soluţii se vor afişa toate. Fişierul de intrare conţine pe fiecare linie cel mult 250 de caractere. Nu se va face distincţie între literă mică şi literă mare. Rezultatul se va afişa pe ecran.

Exemplu:

`INPUT.TXT:`	Pe ecran se va afişa:
`Mama si cu tata merg la piata...`	`A apare de 7 ori`
`Memoriu, memoriu!`	`M apare de 7 ori`

Soluţie:

Rezolvarea problemei presupune contorizarea numărului de apariţie al fiecărei litere în vectorul `a` având indici litere mari şi elemente numere întregi. În acest fel elementul `a['B']=3` codifică 3 apariţii ale literei `B`.

Se iniţializează acest vector cu `0`. Odată cu citirea fiecărei linii din text (în **string**-ul `s`), se va incrementa numărul de apariţii ale fiecărei litere din şir.

După citirea linie cu linie a întregului text se va determina elementul maxim din vectorul `a` şi se vor afişa toate literele care au acest număr de apariţie.

Scanned with OKEN Scanner

```
Program frecventa;
const lit=['a'..'z']+['A'..'Z'];
var s:string;
    a:array['A'..'Z'] of Integer;
    i,max:Integer;
    j:Char;
    f:Text;
Begin
  Assign(f,'INPUT.TXT'); Reset(f);
  for j:='A' to 'Z' do a[j]:=0;
  while not Eof(f) do
  begin
    Readln(f,s);
    for i:=1 to Length(s) do
      if (s[i] in lit)
      then Inc(a[Upcase(s[i])])
  end;
  Close(f);
  max:=a['A'];
  for j:='B' to 'Z' do
    if max<a[j] then max:=a[j];
  for j:='A' to 'Z' do
    if a[j]=max
    then Writeln('litera ', j ,' apare de ',max,' ori')
End.
```

38. Concatenare de cuvinte

Se consideră n cuvinte şi un număr natural (k). Să se determine cel mai lung cuvânt format prin alipirea cuvintelor date care respectă condiţia: nici o literă nu apare în cadrul cuvântului de lungime maximă de mai mult de k ori.

Datele de intrare se găsesc în fişierul INPUT.TXT.

Pe prima linie a fişierului este scris numărul k, iar pe următoarele n linii sunt scrise cele n cuvinte. Cuvântul obţinut ca rezultat va fi scris în fişierul OUTPUT.TXT.

Se va afişa o singură soluţie.

Exemplu:

```
INPUT.TXT:          OUTPUT.TXT:
3                   candpoatefrunza
cand
poate
frunza
pica
toamna
```

Soluţie:

Pentru rezolvarea problemei se va utiliza metoda backtracking; prezentăm o implementare nerecursivă.

Cuvintelele se vor păstra în vectorul `a`, având componente de tip **`string`**. Stiva este reprezentată prin vectorul `st` cu componente de tip întreg, deoarece vor reţine indicii cuvintelor din vectorul `a`, (domeniul în care iau valori elementele vectorului `st` este intervalul `[0,n]`).

Literele în cadrul cuvântului format se vor păstra în vectorul `nr` având indici definiţi pe subdomeniul `'a'..'z'`, iar elementele sunt de tip întreg.

Un nou nivel pe stivă se iniţializează prin depunerea valorii `0`.

Depunerea unui nou element pe un nivel în stivă se realizează în felul următor:
– se decrementează numărul de apariţie al fiecărei litere din cuvântul găsit depus pe acest nivel;
– se incrementează valoarea elementului curent dacă este strict mai mică decât `n`;
– se incrementează numărul de apariţie al fiecărei litere din noul cuvânt depus pe acest nivel.

Validarea unui element depus în stivă necesită verificarea următoarelor condiţii:
– nici o literă nu va avea numărul de apariţie mai mare decât numărul `k`;
– de asemenea se impune condiţia ca elementele din `st` să fie distincte;
– pentru evitarea permutării cuvintelor în cadrul cuvântului format, se impune ca elementele din `st` să apară în ordine crescătoare.

Am obţinut o soluţie dacă nivelul stivei este egal cu `n`. Evident, dacă cele `n` cuvinte pot fi alipite în condiţiile impuse, atunci şirul format are lungime maximă. Dacă nu există această posibilitate, variabila `max` va fi actualizată mereu cu cel mai lung şir obţinut, iar vectorul `st1` va păstra configuraţia stivei care a condus la această lungime. Rezultatele se scriu în fişierul `OUTPUT.TXT`.

```
Program sir_cuvinte;
type stiva=array[1..100] of Integer;
     num=array['a'..'z'] of Integer;
var st,st1:stiva;
    a:array[1..1000] of string[30];
    i,j,n,kk,m,k,ki,max:Integer;
    r:Char; nr:num;
    as,ev,esol:Boolean;
    g,f:Text;

procedure succ(k:Integer; var st:stiva; var nr:num;
var as:Boolean);
var c:string[30]; i:Integer;
begin
  if st[k]<n
  then
```

```
    begin
      if st[k]<>0
      then
        begin
          c:=a[st[k]];
          for i:=1 to Length(c) do Dec(nr[c[i]])
        end;
      Inc(st[k]);
      c:=a[st[k]];
      for i:=1 to Length(c) do Inc(nr[c[i]]);
      as:=true
    end
  else as:=false
end;

procedure valid(k:Integer; st:stiva; nr:num; var ev:Boolean);
begin
  ev:=true;
  for r:='a' to 'z' do if nr[r]>k then ev:=false;
  for i:=1 to k-1 do if st[i]=st[k] then ev:=false;
  if st[k-1]>st[k] then ev:=false
end;

procedure solutie(k:Integer; var esol:Boolean);
var ni:Integer;
begin
  ni:=0;
  for i:=1 to k do ni:=ni+Length(a[st[i]]);
  if ni>max
  then
    begin
      for i:=1 to k do st1[i]:=st[i];
      max:=ni;
      ki:=k
    end;
  esol:=k=n
end;

Begin
  Assign(f,'INPUT.TXT'); Reset(f);
  Readln(f,k);
  n:=0;
  while not Eof(f) do
  begin Inc(n); Readln(f,a[n]) end;
  for r:='a' to 'z' do nr[r]:=0;
  max:=0;
  kk:=1;
  st[kk]:=0;
```

Scanned with OKEN Scanner

```
  while kk>0 do
  begin
    repeat
      succ(kk,st,nr,as);
      if as then valid(kk,st,nr,ev)
    until not as or (as and ev);
    if as
    then
      begin
        solutie(kk,esol);
        if not esol then begin Inc(kk); st[kk]:=0 end
      end
    else Dec(kk)
  end;
  Assign(g,'OUTPUT.TXT'); Rewrite(g);
  for i:=1 to ki do Write(g,a[st1[i]]);
  Close(g)
End.
```

39. Calcule modulo 10

Se consideră fişierul text `INPUT.TXT` care conţine un număr de `n` linii, pe fiecare linie fiind scrise câte `7` cifre. Ele de fapt reprezintă un tablou bidimensional a_{nx7}.

Se cere crearea fişierului text `OUTPUT.TXT` care va conţine tabloul bidimensional b_{nx7} ale cărui elemente sunt determinate astfel: un element `b[i,j]` se calculează ca rest modulo `10` al sumei elementelor din matricea `a` situate pe coloana `j` şi pe linii mai mari sau egale ca `i`.

Exemplu:

```
INPUT.TXT:          OUTPUT.TXT:
2 3 7 2 1 5 4       7 8 9 0 3 2 8
1 7 2 5 5 2 1       5 5 2 8 2 7 4
3 5 6 8 9 4 1       4 8 0 3 7 5 3
1 3 4 5 8 1 2       1 3 4 5 8 1 2
```

Soluţie:

Odată cu citirea tabloului din fişierul de intrare se calculează şi numărul de linii (`n`) al acestuia. Algoritmul de rezolvare al problemei necesită existenţa unei varibile `b` de tip întreg în care se va calcula noua valoare a elementului din tabloul `a`. Acesta va fi tipărit (după transformări) în fişierul de ieşire `OUTPUT.TXT`.

```
Program modulo;
var n,b:Integer;
    a:array[1..1000,1..7] of Byte;
    f,g:Text;
```

Scanned with OKEN Scanner

```
procedure citeste;
var i,j:Integer;
begin
  Assign(f,'INPUT.TXT'); Reset(f); i:=0;
  while not Eof(f) do
  begin Inc(i); for j:=1 to 7 do Read(f,a[i,j]); Readln(f) end;
  n:=i; Close(f)
end;
procedure determin;
var i,j,k:Integer;
begin
  for i:=1 to n do
    for j:=1 to 7 do
    begin
      b:=0;
      for k:=i to n do b:=b+a[k,j];
      a[i,j]:=b mod 10
    end
end;
procedure scrie;
var i,j:Integer;
begin
  Assign(g,'OUTPUT.TXT'); Rewrite(g);
  for i:=1 to n do
  begin
    for j:=1 to 7 do Write(g,a[i,j],' ');
    Writeln(g)
  end;
  Close(g)
end;
Begin
  citeste;
  determin;
  scrie
End.
```

40. Generare de şiruri

Să se determine toate şirurile de 6 numere distincte mai mici decât un număr dat (n≤100) a căror sumă este egală cu numărul s (s≤50). Cele două numere (n şi s) se citesc de la tastatură. Să se determine, de asemenea şi numărul acestor şiruri.

Rezultatele se vor scrie în fişierul de ieşire OUTPUT.TXT sub forma următoare: pe fiecare linie câte o combinaţie de şase numere care au suma egală cu s. Pe ultima linie se va scrie numărul şirurilor.

Scanned with OKEN Scanner

Exemplu:

n=50 s=24

```
OUTPUT.TXT:
1 2 3 4 5 9
1 2 3 4 6 8
1 2 3 5 6 7
3
```

Soluţie:

Pentru rezolvarea problemei se va ultiliza metoda backtracking. Prezentăm o implementare recursivă.

Procedura `back` are doi parametri (`i` de tipul `Byte` şi `s` de tipul `Integer`) reprezentând indicele elementului curent din şirul care se generează, respectiv suma care trebuie acoperită în paşii care urmează.

Pentru a nu genera de două ori un anumit număr ca termen în şir, un termen nou trebuie să fie cel puţin cu `1` mai mare decât termenul anterior. Păstrăm în `sp` valoarea rămasă până la `s` după însumarea celor `i` termeni din şir. Valorile `sp` se depun pe stivă, deoarece fac parte din contextul apelului.

Am obţinut o soluţie atunci când valoarea lui `sp` este `0` şi avem 6 termeni, deoarece în acest moment suma celor 6 elemente ale şirului este egală cu `s`.

```
Program siruri;
var sol:array[0..6] of Byte;
    s:Integer;
    n,nr_sol:Byte;
    g:Text;

procedure scrie;
var i:Byte;
begin
  Inc(nr_sol);
  for i:=1 to 6 do Write(g,sol[i],' ');
  Writeln(g)
end;

procedure back(i:Byte; s:Integer);
var j:Byte; sp:Integer;
begin
  for j:=sol[i-1]+1 to n do
  begin
    sp:=s-j; sol[i]:=j;
    if (i=6) and (sp=0)
    then scrie
    else
      if (sp>0) and (i<6) then back(i+1,sp)
  end
end;
```

Scanned with OKEN Scanner

```
Begin
  Write('n= '); Readln(n);
  Write('s= '); Readln(s);
  Assign(g,'OUTPUT.TXT'); Rewrite(g);
  nr_sol:=0; sol[0]:=0;
  back(1,s);
  Writeln(g,nr_sol); Close(g)
End.
```

41. Ordonare în matrice

Se consideră o matrice a de dimensiune n×m. Să se reaşeze elementele în matrice astfel încât ele să apară în ordine crescătoare atât pe linii cât şi pe coloane. Matricea a se găseşte în fişierul de intrare INPUT.TXT fiecare linie a matricei pe câte o linie din fişier.

Nu este specificată valoarea lui n şi a lui m.

Matricea obţinută se va scrie în mod similar în fişierul OUTPUT.TXT.

În cazul când există mai multe soluţii se va scrie în fişier numai una.

Exemplu:

```
INPUT.TXT:          OUTPUT.TXT:
3 2 5 44 3 1        1 2 2 2 2 2
6 7 55 4 3 2        3 3 3 3 3 4
4 10 4 2 4 5        4 4 4 4 4 5
34 5 6 8 2 4        5 5 5 6 6 7
5 3 2 4 41 3        8 10 34 41 44 55
```

Soluţie:

Pentru rezolvare elementele din matrice sunt ordonate crescător ca într-un şir de n*m elemente şi puse în grupuri de câte m pe fiecare linie. În acelaşi timp se realizează şi corespondenţa între indicii elementelor din matrice şi numărul de ordine al acestora în matricea liniarizată.

```
Program ordine;
var a:array[1..100,1..100] of Integer;
    n,m:Integer;
    f,g:Text;

procedure citeste;
var i,j:Integer;
begin
  Assign(f,'INPUT.TXT'); Reset(f);
  i:=0;
  while not Eof(f) do
  begin
    Inc(i);
```

Scanned with OKEN Scanner

```
    j:=0;
    while not Eoln(f) do
    begin Inc(j); Read(f,a[i,j]) end;
    Readln(f)
  end;
  n:=i; m:=j
end;

procedure scrie;
var i,j:Integer;
begin
  Assign(g,'OUTPUT.TXT'); Rewrite(g);
  for i:=1 to n do
  begin
    for j:=1 to m do Write(g,a[i,j],' ');
    Writeln(g)
  end;
  Close(g)
end;

procedure ordonare;
var i1,j1,i2,j2,k,t,c:Integer;
begin
  for k:=1 to n*m-1 do
    for t:=k+1 to m*n do
    begin
      if k mod m>0 then i1:=k div m+1
                   else i1:=k div m;
      j1:=k-(i1-1)*m;
      if t mod m>0 then i2:=t div m+1
                   else i2:=t div m;
      j2:=t-(i2-1)*m;
      if a[i1,j1]>a[i2,j2]
      then
        begin
          c:=a[i1,j1];
          a[i1,j1]:=a[i2,j2];
          a[i2,j2]:=c
        end
    end
end;

Begin
  citeste;
  ordonare;
  scrie
End.
```

42. Bridge (I)

Într-un joc de bridge există 4 jucători, doi câte doi formând o echipă pe o axă. Celor 4 jucători li se împart în mod egal toate cărţile. Cărţile de joc le vom codifica în felul următor: `2..10,J,D,K,A`. Jucătorii sunt numerotaţi de la 1 la 4. Axele sunt formate din jucătorii numărul 1 şi 3, respectiv jucătorii numărul 2 şi 4.

Fiecare carte considerată onor are un punctaj care nu depinde de culoare. Astfel: pentru o carte codificată cu valoarea `J` se obţine 1 punct, pentru una cu valoarea `D` se obţin 2 puncte, pentru `K` se obţin 3 puncte, iar pentru `A` 4 puncte.

Dându-se o repartiţie de cărţi pentru cei 4 jucători, să se determine care este axa cu cele mai multe puncte şi câte puncte au împreună cei doi jucători.

Fişierul de intrare `INPUT.TXT` conţine 16 linii. Fiecare grup de 4 linii conţine cărţile primite, în ordine, de fiecare jucător (de la 1 la 4). Pe fiecare linie din cele patru sunt scrise numai cărţi de o singură culoare. Dacă un jucător nu primeşte nici o carte de o anumită culoare atunci pe una din cele patru linii care-i corespund va fi trecut caracterul `F`.

Exemplu:

`INPUT.TXT:`

```
345678
DKA
F
JDKA
2A
234J
23410JD
234
910D
78910
KA
567
JK
56
56789
8910
```

Se va afişa pe ecran:

```
Axa jucatorilor 1 3 are punctajul 28
```

Soluţie:

Datele din fişierul de intrare se citesc într-o variabilă de tip **`string`**. Odată cu această citire se calculează punctele fiecărui jucător, punctajele fiind memorate într-un vector de dimensiune 4. Se calculează, apoi se afişează punctajele pe axa `a1` şi axa `a2`.

```
Program bridge;
var a:array[1..4] of Byte;
    a1,a2,i,j,k:Integer;
    b:string[50]; f:Text;
```

Scanned with OKEN Scanner

```
Begin
  Assign(f,'INPUT.TXT'); Reset(f);
  for i:=1 to 4 do
  begin
    a[i]:=0;
    for j:=1 to 4 do
    begin
      Readln(f,b);
      if b<>'F'
      then
        for k:=1 to Length(b) do
         case b[k] of
            'J':a[i]:=a[i]+1;
            'D':a[i]:=a[i]+2;
            'K':a[i]:=a[i]+3;
            'A':a[i]:=a[i]+4
          end
    end
  end;
  Close(f);
  a1:=a[1]+a[3]; a2:=a[2]+a[4];
  if a1>a2
  then Writeln('Axa jucatorilor 1 3 are punctajul ',a1)
  else if a1<a2
       then Writeln('Axa jucatorilor 2 4 are punctajul ',a2)
       else Writeln('Axele au acelasi punctaj ',a1)
End.
```

43. Bridge (II)

Într-un joc de bridge există 4 jucători, doi câte doi jucând împreună pe o axă. Le sunt împărţite în mod egal toate cărţile. `2..10,J,D,K,A` sunt codurile pentru valorile cărţilor. Jucătorii sunt numerotaţi de la 1 la 4. Axele sunt formate din jucătorii având numărul 1 şi 3, respectiv jucătorii cu numărul 2 şi 4. Onoruri sunt considerate cărţile `J,D,K,A`. Cărţile pot avea 4 culori, codificate prin caracterele: `1, 2, 3, 4`.

Dându-se o repartiţie de cărţi pentru cei 4 jucători, să se determine: axa cu cele mai multe cărţi de aceeaşi culoare, şi culoarea respectivă; câte cărţi de această culoare are această axă; câte onoruri are axa din această culoare.

Fişierul de intrare `INPUT.TXT` conţine 16 linii. Fiecare grup de 4 linii conţine cărţile primite în ordine de fiecare jucător (de la 1 la 4).

Pe fiecare linie din cele patru sunt scrise numai cărţi de o singură culoare. Codul culorii este primul caracter de pe acea linie. Dacă un jucător nu primeşte nici o carte de o anumită culoare atunci pe una din cele patru linii, după codul culorii respective, va fi trecut caracterul `'F'`. Rezultatele vor fi afişate pe ecran.

Scanned with OKEN Scanner

Exemplu:

```
INPUT.TXT:
1345678
2DKA
3F
4JDKA
12A
2234J
323410JD
4234
1910D
278910
3KA
4567
1JK
256
356789
48910
```

Pe ecran se va afişa:

```
Axa jucatorilor 2-4 au din culoarea 4
un numar de 11 carti cu 2 onoruri.
```

Soluție:

Se determină (pe cele două axe de jucători), numărul de cărți din fiecare culoare. În acest scop se foloseşte o matrice `L` de dimensiune `4x4` în care elementul `L[i,j]` are semnificația: din culoarea `i` jucătorul `j` are `L[i,j]` cărți.

Datele din fişierul de intrare se citesc într-o variabilă `b` de tip **`string`**.

Elementul `L[i,j]` este egal cu numărul de caractere al variabilei `b`, mai puțin primul caracter, care reprezintă codul culorii, iar dacă în şirul `b` apare cartea de valoare `10` atunci `L[i,j]` va fi egal cu lungimea lui `b` minus `2`.

În mod analog se calculează numărul de onoruri al fiecărui jucător din fiecare culoare. În matricea `O` de dimensiune `4x4` elementul `O[i,j]` reprezintă numărul de onoruri din culoarea `i` al jucătorului `j`. În final se însumează elementele din cele două matrice pe coloanele `1` şi `3` respectiv `2` şi `4` şi se calculează maximele.

```
Program bridge;
const onor=['A','D','J','K'];
var L,O:array[1..4,1..4] of Byte;
    a1,a2,m1,m2,c1,c2,ax,c,max,i,v,e,j,k,on:Integer;
    b:string[50];
    f:Text;
Begin
  for i:=1 to 4 do
    for j:=1 to 4 do
    begin L[i,j]:=0; O[i,j]:=0 end;
  Assign(f,'INPUT.TXT'); Reset(f);
```

Scanned with OKEN Scanner

```
  for j:=1 to 4 do
    for c:=1 to 4 do
    begin
      Readln(f,b);
      if b[2]<>'F'
      then
        begin
          Val(b[1],v,e);
          if Pos('10',b)<>0 then L[v,j]:=Length(b)-2
                            else L[v,j]:=Length(b)-1;
          for k:=2 to Length(b) do
            if b[k] in onor then Inc(O[v,j])
        end
    end;
  Close(f);
  a1:=0; a2:=0;
  for i:=1 to 4 do
  begin
    m1:=L[i,1]+L[i,3];
    if m1>a1
    then begin a1:=m1; c1:=i end
  end;
  for i:=1 to 4 do
  begin
    m2:=L[i,2]+L[i,4];
    if m2>a2
    then begin a2:=m2; c2:=i end
  end;
  if a1>a2
  then begin max:=a1; c:=c1; ax:=1 end
  else begin max:=a2; c:=c2; ax:=2 end;
  on:=O[c,ax]+O[c,ax+2];
  Writeln('Axa jucatorilor ',ax,'-',ax+2,' are din culoarea ',c);
  Writeln(' un numar de ',max,' carti cu ',on,' onoruri')
End.
```

44. Sortare prin interclasare

Se dă un şir (a_n), $1<n<100$, cu elemente numere reale. Să se ordoneze descrescător acest şir, folosind algoritmul sortării prin interclasare (metoda "divide et impera").

Datele de intrare se citesc dintr-un fişier text SIR.TXT cu structura:

a1 a2 ... an

Elementele şirului sunt separate printr-un spaţiu. Şirul sortat se va memora într-un fişier ORDONAT.TXT, pe o singură linie, elementele fiind separate printr-un spaţiu.

Scanned with OKEN Scanner

Exemplu:

```
SIR.TXT:                    ORDONAT.TXT:
2 1 3 7 5.2 6.123           7 6.123 5.2 3 2 1
```

Soluţie:

Se va utiliza metoda "divide et impera" care presupune împărţirea şirului iniţial în două subşiruri de lungimi aproximativ egale. Vom repeta împărţirea în subşiruri până când subşirurile obţinute vor fi de lungime 1. Un subşir de lungime 1 este ordonat (crescător sau descrescător). Subşirurile ordonate vor fi interclasate, constituind noi subşiruri ordonate. Procedeul se repetă până când se obţine soluţia problemei iniţiale.

```
Program sortare_prin_interclasare;
var n,i:Byte;
    el:Real;
    a:array[1..100] of Real;
    f,g:text;

procedure sortintc(u,v:Byte);
var m:Byte;
  procedure intercl;                    { procedura de interclasare }
  var b:array[1..100] of Real;
      i,j,k,m:Byte;
  begin
    m:=(u+v) div 2;          { Se imparte sirul in doua subsiruri }
    i:=u; j:=m+1;
    k:=u-1;
    repeat                    { Se alege elementul mai mare si se }
      k:=k+1;                 { memoreaza in sirul b. Se trece la }
      if a[i]>a[j]            { elementul urmator in subsirul din }
      then                    { care s-a preluat elementul, pana  }
        begin                 { la terminarea unuia dintre cele   }
          b[k]:=a[i];                                { doua subsiruri }
          i:=i+1
        end
      else begin b[k]:=a[j]; j:=j+1 end;
    until (i>m) or (j>v);
    if i>m                   { se trec in b elementele ramase din }
    then                     { subsirul netransferat complet in b }
      for i:=j to v do
      begin k:=k+1; b[k]:=a[i] end
    else
      for j:=i to m do
      begin k:=k+1; b[k]:=a[j] end;
                    { se copiaza subsirul rezultat din b in a }
    for i:=u to v do a[i]:=b[i]
  end;
```

Scanned with OKEN Scanner

```
begin                          { corpul procedurii sortintc      }
  if u<v
  then
    begin
      m:=(u+v) div 2;          { impartire în subsiruri          }
      sortintc(u,m);           { sortarea primului subsir        }
      sortintc(m+1,v);         { sortarea subsirului al doilea   }
      intercl                  { inteclasarea sirurilor ordonate }
    end
end;

Begin
  Assign(f,'SIR.TXT');
  Reset(f);
  i:=0;                    { citire sir din fisierul de intrare }
  repeat
    i:=i+1;
    Read(f,a[i]);
  until Eoln(f);
  Close(f);
  n:=i;
  Write(' Sirul initial : ');          { afisare sir initial }
  for i:=1 to n do
    Write(a[i]:1:3,' ');
  Writeln;
  sortintc(1,n);                   { apel procedură de sortare }
  Assign(g,'ORDONAT.TXT');         { memorare sir sortat în    }
  Rewrite(g);                      { fisierul de iesire        }
  for i:=1 to n-1 do Write(g,a[i]:1:3,' ');
  Write(g,a[n]:1:3);
  Close(g);
  Reset(g);
  Write(' Sirul sortat : ');
  repeat                   { afisare sir din fisierul de iesire }
    Read(g,el);
    Write(el:1:3,' ');
  until Eoln(g);
  Close(g);
  Readln
End.
```

45. Interclasare

Se dau două şiruri de numere întregi, ordonate: A şi B cu m respectiv n elemente. Să se determine un al treilea şir C obţinut prin interclasarea elementelor şirurilor A şi B.

Datele de intrare se vor citi dintr-un fişier text.

Scanned with OKEN Scanner

Soluţie:

Prezentăm o soluţie implementată în limbajul C:

```
#include<conio.h>
#include<stdio.h>
FILE *f,*g;
int n,i,j,k,m,a[20],b[20],c[40];
void main()
{
  f=fopen("f1.txt","rb");
  fscanf(f,"%d",&n);
  fscanf(f,"%d",&m);
  for(i=1;i<=n;i++) fscanf(f,"%d",&a[i]);
  for(i=1;i<=m;i++) fscanf(f,"%d",&b[i]);
  fclose(f);
  k=0;
  i=n;
  j=m;
  while(m!=0&&n!=0) {
    k++;
    if(a[i-n+1]<=b[j-m+1]) {c[k]=a[i-n+1]; n--; }
      else {c[k]=b[j-m+1]; m--; }
    if(m==0) for(j=i-n+1,n=0;j<=i;j++) c[++k]=a[j];
    if(n==0) for(i=j-m+1,m=0;i<=j;i++) c[++k]=b[i];
  }
  g=fopen("f2.txt","wb");
  fprintf(g,"Sirul sortat prin interclasare:\n");
  for(i=1;i<=k;i++) fprintf(g,"%3d",c[i]);
  fclose(g);
}
```

46. Şirul lui Fibonacci

Să se citească un număr natural n, n>0 şi, dacă numărul nu face parte din şirul lui Fibonacci, să se descompună în mod unic în sumă de numere Fibonacci.

Numărul n se citeşte de la tastatură, rezultatele se afişează pe ecran.

Exemple:

n=13	Pe ecran se va afişa: n = numar Fibonacci
n=341	Pe ecran se va afişa: n = 233+89+13+5+1

Soluţie:

Unicitatea descompunerii este asigurată de faptul că termenii descompunerii sunt generaţi în ordine descrescătoare.

Scanned with OKEN Scanner

```
Program Descompunere_n_in_suma_de_numere_Fibonacci;
type Natural=0..Maxint;
var n:Natural;                    { numarul care se descompune  }
    a,b,c:Natural;                { trei termeni consecutivi ai }
                                  { sirului Fibonacci           }
    Fibo:Boolean; { arata daca n este din sirul lui Fibonacci }
Begin
  Write('Introduceti numarul n>0 , n='); Read(n);
  Write('n = ');          { in timpul executiei n se distruge }
  Fibo:=true;
  repeat
    a:=0; b:=1; c:=a+b;
    while c<n do                  { se calculeaza cel mai mic }
    begin                         { termen din sirul lui      }
      a:=b; b:=c;                 { Fibonacci, mai mare sau   }
      c:=a+b                      { egal cu n }
    end;
    if c=n
    then
      begin
        if Fibo then Writeln('numar Fibonacci')
                else Writeln(c);
        n:=0
      end
    else begin Fibo:=false; Write(b,'+'); n:=n-b end
  until n=0
End.
```

47. Putere calculată rapid

Fiind dați x>0 real şi n>6 natural, se cere să se calculeze puterea x^n efectuând mai puțin de n-1 înmulțiri. Baza şi exponentul se citesc de la tastatură, iar pe ecran se vor afişa puterea calculată şi numărul de înmulțiri efectuate.

Exemplu:

```
baza=2                    Pe ecran se va afişa:
exponent=9                Puterea=2^9=512
                          Numar de inmultiri=5
```

Soluție:

Se foloseşte un algoritm care se bazează pe exprimarea exponentului n în baza 2, de unde se deduce că puterea x^n se poate scrie ca un produs de puteri de forma x^{2^k}, k>0. Puterile de forma x^{2^k}, k>0 se pot calcula rapid prin ridicări succesive la pătrat.

Exemplu:

$$x^9 = x^{1001_2} = x^8 \cdot x = \left(\left(x^2\right)^2\right)^2 \cdot x$$

Scanned with OKEN Scanner

```
Program Putere_calculata_rapid;
type Natural=0..100;
var exponent,numar_inmultiri:Natural;
    baza,putere:1..Maxlongint;

Begin
  Write('baza='); Readln(baza);
  Write('exponent='); Readln(exponent);
  Write('Puterea='); Write(baza,'^',exponent,'=');
  putere:=1;
  numar_inmultiri:=0;
  while exponent>0 do
  begin
    if Odd(exponent)
    then
      begin
        putere:=putere*baza;
        numar_inmultiri:=numar_inmultiri+1
      end;
    baza:=Sqr(baza);
    numar_inmultiri:=numar_inmultiri+1;
    exponent:=exponent div 2
  end;
  Writeln(putere);
  Writeln('Numar de inmultiri=',numar_inmultiri-1)
End.
```

48. Conversie din arab în roman

Să se citească de la tastatură un număr natural n, n<5000 şi să se afişeze pe ecran acest număr scris cu cifre romane.

Exemplu:

n =2785 Pe ecran se va afişa: MMDCCLXXXV

Soluţie:

Se va folosi o variabilă de tip enumerat, ordin, care arată dacă valoarea numărului este de ordinul sutelor, zecilor sau unităţilor.

```
Program Conversie_din_scriere_araba_in_scriere_romana;
var n,v9,v5,v4,v1:1..4999;          { numar, variabile de lucru }
    m,d,c:'C'..'X';                 { cifre romane              }
    ordin:(unitati,zeci,sute);

Begin
  Write('n='); Readln(n);
  while n>=1000 do
  begin n:=n-1000; Write('M') end;
```

Scanned with OKEN Scanner

```
  v9:=900; v5:=500; v4:=400; v1:=100; ordin:=sute;
  while n>0 do
  begin
    case ordin of
         sute:begin m:='M'; d:='D'; c:='C' end;
         zeci:begin m:='C'; d:='L'; c:='X' end;
      unitati:begin m:='X'; d:='V'; c:='I' end
    end;
    case n div v1 of        { daca prima cifra din n ramas este }
         9:begin n:=n-v9; Write(c,m) end;
         4:begin n:=n-v4; Write(c,d) end;
      5..8:begin n:=n-v5; Write(d) end
    end;
    while n>=v1 do                          { prima cifra 1, 2, 3 }
    begin n:=n-v1; Write(c) end;
    v9:=v9 div 10; v5:=v5 div 10;
    v4:=v4 div 10; v1:=v1 div 10;
    if ordin>unitati then ordin:=Pred(ordin)
  end;
  Writeln
End.
```

49. Pătrat magic

Să se verifice dacă elementele unei matrice pătratice $A_{n\times n}=(a_{ij})_{\substack{i=\overline{1,n}\\ j=\overline{1,n}}}$, $a_{ij}\in N^*$ reprezintă un pătrat magic. Elementele matricei reprezintă un pătrat magic dacă sumele elementelor din fiecare linie, coloană şi de pe cele două diagonale sunt egale. Valoarea comună a tuturor acestor sume se numeşte *sumă magică*. Se presupune că datele de intrare sunt corecte. Elementele matricei se citesc de la tastatură. Dacă acestea constituie un pătrat magic, se va afişa pe ecran un mesaj corespunzător şi suma magică.

Exemplu:

```
16  3  2 13
 5 10 11  8
 9  6  7 12
 4 15 14  1
```

Pe ecran se va afişa:

```
Patrat magic
34
```

Soluţie:

Când se parcurg elementele unei coloane j: $a_{1j}, a_{2j}, \ldots, a_{nj}$, indicele coloanei rămâne fix, iar indicele de linie variază de la 1 la n.

Diagonala principală a unei matrice pătratice cuprinde elementele $a_{11}, a_{22}, \ldots, a_{nn}$, deci elementele cu proprietatea că indicele de linie este egal cu cel de coloană; prin urmare diagonala principală cuprinde elementele a_{ii}, $i=\overline{1,n}$.

Scanned with OKEN Scanner

Diagonala secundară cuprinde elementele $a_{1,n}, a_{2,n-1}, \ldots, a_{n1}$, adică elementele $a_{i,n-i+1}$, $i=\overline{1,n}$ (suma dintre indicele de linie şi cel de coloană este n+1.

Suma magică este notată cu Sm, iar suma curentă este notată cu S.

Programul se opreşte forţat în momentul în care se constată că pătratul nu este magic.

```
Program Patrat_magic;
var a:array[1..20,1..20] of Integer;
    S,Sm,n,i,j:Byte;

Begin
  Write('n='); Readln(n);
  Writeln('Dati matricea:');      { se solicita toata matricea }
  for i:=1 to n do
    for j:=1 to n do Read(a[i,j]);
  Sm:=0;                          { Sm = suma pe diagonala principala }
  for i:=1 to n do Sm:=Sm+a[i,i];
  S:=0;                           { S = suma pe diagonala secundara }
  for i:=1 to n do S:=S+a[i,n-i+1];
  if Sm=S
  then
    begin
      for i:=1 to n do
      begin
        S:=0;                                  { S = suma pe linia "i" }
        for j:= 1 to n do S:=S+a[i,j];
        if Sm<>S
        then
          begin
            Writeln('Nu este patrat magic'); Halt
          end
      end;
    for j:=1 to n do
    begin
      S:=0;                                    { S=suma pe coloana "j" }
      for i:= 1 to n do S:=S+a[i,j];
      if Sm<>S
      then
        begin
          Writeln('Nu este patrat magic'); Halt
        end
    end;
    Writeln('Patrat magic'); Writeln(Sm)
  end
  else Writeln('Nu este patrat magic')
End.
```

Scanned with OKEN Scanner

50. Loto

Să se simuleze jocul LOTO, cunoscând numărul de participanţi p, numărul m de numere extrase din urnă (m≤10) şi numărul L de numere cuprinse între 1 şi 99 care se trec pe un bilet LOTO (L fixat, L≤9). Pentru fiecare participant la joc, se cunosc: numele, numărul biletului LOTO, cele L numere trecute pe bilet şi numerele extrase din urnă. (C.S.P., Bucureşti, 1984)

Soluţie:

Programul afişează participanţii în ordine descrescătoare a numărului de numere ghicite. La acelaşi punctaj are prioritate cel care a ghicit numere situate mai aproape de începutul extragerii. Pentru fiecare participant se afişează numele, numărul biletului LOTO şi numerele alese, cele ghicite sunt puse între ghilimele.

Pentru calculul distanţei faţă de începutul extragerii, se însumează poziţiile pe care se află în extragere numerele ghicite de către participant.

```
Program Simulare_joc_loto;
type numar_loto=1..99;
     Articol=record
               nume:string[30];
               numar_bilet:Integer;
               numar:array[1..9] of numar_loto;
                                    { continutul biletului }
               ghicit:array[1..9] of Boolean;
               punctaj,distanta:Integer
             end;
var participant:array[1..100] of Articol;
    aux:Articol;
    numar_extras:array[1..20] of numar_loto;
    p,m,L,i,j,k:Integer;
    ordonat:Boolean;
Begin
  Write('Introduceti numarul de participanti p='); Readln(p);
  Write('Introduceti numarul de numere extrase m='); Readln(m);
  Write('Introduceti numarul de numere de pe un bilet LOTO L=');
  Readln(L);
  Write('Numerele extrase din urna : ');
  for i:=1 to m do Read(numar_extras[i]);
  Readln;
  for i:=1 to p do
    with participant[i] do
    begin
      Write('Numele participantului : '); Readln(nume);
      Write('Numarul biletului : '); Readln(numar_bilet);
      punctaj:=0; distanta:=0;
```

Scanned with OKEN Scanner

```
      Write('Numerele de pe bilet : ');
      for j:=1 to L do
      begin
        Read(numar[j]);
        ghicit[j]:=false;
        for k:=1 to m do
          if numar_extras[k]=numar[j]
          then
            begin
              punctaj:=punctaj+1;
              distanta:=distanta+k;
              ghicit[j]:=true
            end
      end;
      Readln
    end;
  repeat
    ordonat:=true;
    for i:=1 to p-1 do
      with participant[i] do
      begin
        if (punctaj < participant[i+1].punctaj) or
           (punctaj = participant[i+1].punctaj) and
           (distanta > participant[i+1].distanta)
        then
          begin
            aux:=participant[i];
            participant[i]:=participant[i+1];
            participant[i+1]:=aux;
            ordonat:=false
          end
      end
  until ordonat;
  Write('Numele Participantului','Numar bilet':15);
  Writeln(' Numerele de pe bilet');
  for i:=1 to 58 do Write('*'); Writeln;
  for i:=1 to p do
    with participant[i] do
    begin
      Write(nume,' ':26-Length(nume),numar_bilet:8,'     ');
      for j:=1 to L do
        if ghicit[j] then Write('"',numar[j]:2,'" ')
                     else Write(' ',numar[j]:2,'  ');
      Writeln
    end
End.
```

Scanned with OKEN Scanner

51. Relaţii de echivalenţă

Se citesc n perechi de elemente notate cu litere mici din alfabetul latin, care se găsesc într-o *relaţie de echivalenţă* R. Perechile sunt separate prin caracterul `'R'`. Se cere să se afişeze clasele de echivalenţă ale relaţiei R.

Exemplu:

```
n=6
aRb
cRd
eRf
bRf
gRg
cRh
```

Pe ecran se va afişa:

```
Clasa  1 : a b e f
Clasa  2 : c d h
Clasa  3 : g
```

Soluţie:

Fiind dată o mulţime *M*, spunem că *R* este o *relaţie de echivalenţă*, dacă au loc proprietăţile de *reflexivitate, simetrie* şi *tranzitivitate.* O clasă de echivalenţă este cea mai mare submulţime din *M* care cuprinde elemente echivalente între ele. Clasele de echivalenţă sunt disjuncte, iar reunite dau mulţimea *M*.

Programul este realizat în două variante:

Varianta 1:

– clasele se construiesc treptat, pe măsură ce se citesc perechile de elemente x,y;

– se caută clasa de echivalenţă căreia îi aparţine perechea citită `x,y`, adăugând perechea clasei respective;

– dacă `x` aparţine unei clase, iar `y` nu aparţine claselor existente, se adaugă `y` clasei lui `x`;

– dacă `x` aparţine unei clase, iar `y` aparţine altei clase, se reunesc cele două clase în prima, iar a doua se transformă în mulţimea vidă;

– dacă `x,y` nu aparţin nici unei clase dintre cele construite, vor forma o nouă clasă;

```
Program Clase_de_echivalenta_solutia_1;
var clasa:array[1..40] of set of 'a'..'z';
    x,y:'a'..'z'; r:Char;
    n,nrc,i,j,k:Integer;    { n=numar perechi, nrc=numar clase }
    a:array[1..2] of 1..40;
                  { indicii claselor carora le apartin x si y }
Begin
  Write('Introduceti numarul de perechi n='); Readln(n);
  nrc:=0;                                  { initial nici o clasa }
  for i:=1 to n do
  begin
    Write('Perechea x,y:'); Readln(x,r,y);
    k:=0;
```

```
    for j:=1 to nrc do
      if (x in clasa[j]) or (y in clasa[j])
      then begin k:=k+1; a[k]:=j end;
      case k of
                                    { x,y formeaza o noua clasa }
     0 : begin nrc:=nrc+1; clasa[nrc]:=[x,y] end;
                  { x sau y apartine clasei cu indicele a[1] }
     1 : clasa[a[1]]:=clasa[a[1]]+[x,y];
       { x,y apartin unor clase diferite cu indicii a[1],a[2] }
     2 : begin clasa[a[1]]:=clasa[a[1]]+clasa[a[2]];
           if a[2]=nrc then nrc:=nrc-1
                       else clasa[a[2]]:=[]
         end
    end
  end;
  k:=0;
  for i:=1 to nrc do
    if clasa[i]<>[]
    then
      begin
        k:=k+1;
        Write('Clasa',k:3,' :');
        for x:='a' to 'z' do
          if x in clasa[i] then Write(x:2);
        Writeln
      end
End.
```

Varianta 2:

– fiecare pereche citită va forma iniţial câte o clasă;

– clasele cu elemente comune se reunesc în clasa cu indice mai mare, cealaltă transformându-se în mulţimea vidă; rămân, în final, doar clase disjuncte care reprezintă chiar clasele de echivalenţă şi "clasele" vide.

```
Program Clase_de_echivalenta_solutia_2;
var clasa:array[1..40] of set of 'a'..'z';
    x,y:'a'..'z'; r:Char;
    n,i,j,k:0..40;

Begin
  Write('Introduceti numarul de perechi n='); Readln(n);
  for i:=1 to n do
  begin
    Readln(x,r,y);
    clasa[i]:=[x,y]
  end;
```

Scanned with OKEN Scanner

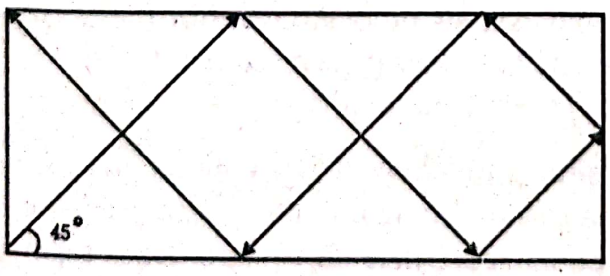
```
  for i:=1 to n-1 do
  begin
    for j:=i+1 to n do
      if clasa[i]*clasa[j]<>[]
      then                        { clasele nedisjuncte se reunesc }
        begin
          clasa[j]:=clasa[j]+clasa[i];
          clasa[i]:=[];                          { devine vida }
          Break                  { iesire fortata din for }
        end;
  end;
  k:=0;
  for i:=1 to n do
    if clasa[i]<>[]
    then
      begin
        k:=k+1;
        Write('Clasa',k:3,' :');
        for x:='a' to 'z' do
          if x in clasa[i] then Write(x:2);
        Writeln
      end
End.
```

52. Raza de lumină

Fie o matrice `A` cu `m` linii şi `n` coloane (`m, n≤256`). O rază porneşte din colţul stânga jos al matricei, formând un unghi de 45° cu prima coloană şi ultima linie. Raza se reflectă când ajunge pe prima sau ultima linie, respectiv prima sau ultima coloană. Să se afişeze toate poziţiile atinse de rază, până când ea ajunge într-un colţ al acesteia.

Soluţie:

Indicii elementelor din matrice reprezintă poziţiile prin care trece raza. Prin urmare avem nevoie de matrice numai pentru a gândi problema.



Când raza:
1) urcă – indicele de linie scade
2) coboară – indicele de linie creşte
3) se deplasează spre dreapta – indicele de coloană creşte
4) se deplasează spre stânga – indicele de coloană scade

Scanned with OKEN Scanner

a) reflexia de linie are loc atunci când se întâlneşte linia 1 sau m
– determină: – creşterea liniei (dacă anterior a scăzut)
– scăderea liniei (dacă anterior a crescut)

b) reflexia de coloană are loc dacă se întâlneşte coloana 1 sau n
– determină: – creşterea coloanei (dacă anterior a scăzut)
– scăderea coloanei (dacă anterior a crescut)

Creşterea, respectiv scăderea liniei/coloanei se realizează prin adunarea valorii +1 sau -1 la indicele de linie/coloană. Rezultă că simularea fenomenului de reflexie se va realiza prin schimbarea semnului valorii +1/-1 care se adună indicelui de linie/coloană.

```
Program Raza_de_lumina;
var m,n,i,j:Byte;          {   rl=simuleaza deplasarea pe linie }
    rl,rc:Shortint;        { rc=simuleaza deplasarea pe coloana }
    np:0..Maxlongint;      {    np=numarul de pozitii ale razei }

Begin
  Write('m,n='); Readln(m,n);
  i:=m; j:=1;              {  raza pleaca din coltul stinga jos }
  rl:=-1; rc:=1;
  np:=0;
  Writeln('Pozitiile prin care trece raza de lumina:');
  repeat
    Write(i,'-',j,'  ');
    i:=i+rl; j:=j+rc;
    if i in [1,m] then rl:=-rl; {   raza se reflecta de linie }
    if j in [1,n] then rc:=-rc; { raza se reflecta de coloana }
    np:=np+1;
    if np mod 10 = 0 then Writeln;
  until (i in [1,m]) and (j in [1,n]); { pana ajunge intr-un colt }
  Writeln(i,'-',j)
End.
```

53. *Interclasare cu santinele*

Se cere să se interclaseze două şiruri de numere reale, ordonate crescător (descrescător). Prin interclasare se înţelege crearea unui nou şir ordonat crescător (descrescător), format din elementele şirurilor date ordonate crescător (descrescător).

Soluţie:

Dacă a(m)<b(n), se poate introduce în şirul a un element fictiv numit *santinelă*, egal cu b(n)+1, care va determina terminarea şirului a numai după ce s-au preluat cele m+n elemente în şirul c; în acest fel se elimină secvenţele de copiere. Dacă b(n)< a(m), santinela a(m)+1 trebuie introdusă la sfârşitul şirului b, iar dacă a(m)=b(n) trebuie introdusă câte o santinelă la sfârşitul fiecărui şir. Pentru a nu mai testa aceste cazuri, este mai simplu să se introducă de la început santinele în ambele şiruri.

Scanned with OKEN Scanner

```
Program Interclasare_cu_santinele;
var a,b,c:array[1..100] of Integer;
    n,m,i,j,k:Integer;
Begin
  Write('n='); readln(m);
  Write('n='); readln(n);
  Write('Introduceti sirul ordonat a :');
  for i:=1 to m do Read(a[i]);
  Write('Introduceti sirul ordonat b :');
  for i:=1 to n do Read(b[i]);
  a[m+1]:=b[n]+1; b[n+1]:=a[m]+1;       { crearea santinelelor }
  i:=1; j:=1;                                  { interclasarea }
  for k:=1 to m+n do
    if a[i]<=b[j] then begin c[k]:=a[i]; i:=i+1 end
                  else begin c[k]:=b[j]; j:=j+1 end;
  Writeln('Sirul obtinut prin interclasare :');
  for k:=1 to m+n do Write(C[k]:4);
  Readln
End.
```

54. Produs

Să se afişeze coeficienţii polinomului:

$(X+a_1)(X+a_2)\ldots(X+a_n)$, $n\in N^*$, $a_i\in R$, $i=1,\ldots,n$.

Soluţie:

Rezolvarea problemei presupune simularea prin program a calculului matematic de mai sus. Polinomul care trebuie determinat are gradul n şi se poate scrie:

$P(X) = X^n + c_1X^{n-1} +\ldots+ c_n$.

Dacă am calcula manual am face înmulţirile pe rând. La fel vom proceda şi în program. Vom pleca de la un polinom de grad 0: $P(X)=1$ şi îl vom înmulţi pe rând cu $(X+a_1)$, apoi polinomul obţinut îl vom înmulţi cu $(X+a_2)$ ş.a.m.d. până la înmulţirea cu $(X+a_n)$. Va trebui să deducem formulele coeficienţilor polinomului care se obţine prin înmulţirea cu un monom oarecare, de exemplu $(X+a_i)$.

Când înmulţim cu $(X+a_1)$ gradul polinomului anterior este 0, când înmulţim cu $(X+a_2)$ gradul polinomului anterior este 1 ş.a.m.d., când vom înmulţi cu $(X+a_i)$ gradul polinomului anterior este $i-1$

Identificând coeficienţii, obţinem:

$c_0 = 1$; $c_j = b_j + a_ib_{j-1}$, $j=1,\ldots,i-1$, $c_i = a_ib_{i-1}$.

Dacă $b_i = 0$ atunci putem scrie: $c_0 = 1$; $c_j = b_j + a_ib_{j-1}$, $j=1,\ldots,i$.

Deoarece la fiecare înmulţire este nevoie de un nou b_i egal cu 0 îi vom iniţializa pe toţi de la început cu 0.

Scanned with OKEN Scanner

```
Program Calculul_coeficientilor_polinomului_P;
var a,b,c:array[0..20] of Integer;
    n,i,j:Byte;
Begin
  Write('n='); Readln(n);
  Write('Dati coeficientii a[i],i=1,',n,' : ');
  for i:=1 to n do Read(a[i]);
  for i:=1 to n do b[i]:=0;
  b[0]:=1; c[0]:=1;
  for i:=1 to n do
  begin
    for j:=1 to i do c[j]:=b[j]+a[i]*b[j-1];
    b:=c
  end;
  Write('Coeficientii polinomului: ');
  for i:=0 to n do Write(c[i],' ');
  Writeln
End.
```

55. *Operații cu mulțimi*

Se citesc două mulțimi A şi B, ale căror elemente sunt notate cu litere mici din alfabet. Se afişează elementele mulțimilor A∪B, A∩B şi A\B.

Soluție:

```
Program Operatii_cu_multimi;
type multime=set of 'a'..'z';
var A,B:multime;
procedure citire_multime(var M:multime);
var elem:'a'..'z';
begin M:=[];
  while not Eoln do begin Read(elem); M:=M+[elem] end;
  Readln
end;
procedure Scriere_multime(M:multime);
var elem:'a'..'z';
begin
  if M=[] then Writeln('Multimea vida')
          else
            begin
              for elem:='a' to 'z' do
                if elem in M then Write(elem:2);
              Writeln
              end
end;
```

Scanned with OKEN Scanner

```
Begin
  Write('Dati multimea A :'); Citire_multime(A);
  Write('Dati multimea B :'); Citire_multime(B);
  Write('Reuniunea multimilor A si B este ');
  Scriere_multime(A+B);
  Write('Intersectia multimilor A si B este ');
  Scriere_multime(A*B);
  Write('Diferenta multimilor A si B este ');
  Scriere_multime(A-B)
End.
```

56. Cuvinte de lungime maximă

Să se determine cuvintele de lungime maximă dintr-o linie de intrare. Cuvintele sunt separate prin separatorii uzuali: spaţiu, virgulă, semnul exclamării etc.

Soluţie:

Deoarece pot exista mai multe cuvinte cu aceeaşi lungime maximă, cuvintele trebuie memorate într-un tablou.

Se detectează începutul unui nou cuvânt: apare un caracter care nu este separator, după un separator. Pentru a întâlni o asemenea situaţie în cazul primului, respectiv ultimului cuvânt din linie trebuie să adăugăm liniei unul sau două spaţii:

a) un spaţiu la începutul liniei – pentru primul cuvânt din linie care ar putea să nu apară după spaţii;

b) un spaţiu atât la începutul cât şi la sfârşitul liniei – pentru cazul când linia începe cu un singur spaţiu care ar trebui comparat cu elementul anterior lui şi pentru ultimul cuvânt din linie care s-ar putea să nu aibă spaţiu după el.

```
Program Cuvintele_de_lungime_maxima_din_linia_de_intrare;
var linie:string;                              { linia de intrare }
    cuv:array[1..30] of string[20];
    lmax,i,n:Byte;
procedure Extrage_cuvintele_din_linia_de_intrare;
const sep=[' ',',','.','!',':',';','?'];{multimea separatorilor }
var i:Byte;
begin
  linie:=' '+linie;                          { simplifica algoritmul }
  n:=0;                                      { n=numarul de cuvinte  }
  for i:=2 to Length(linie) do
    if not ( linie[i] in sep )    { linie[i]= un car. din cuv. }
    then begin
           if  linie[i-1] in sep          { incepe un nou cuvant }
       then begin n:=n+1; cuv[n]:='' end;
       cuv[n]:=cuv[n]+linie[i]
     end
end;
```

```
Begin
  Writeln('Linia de intrare:'); Readln(linie);
  Extrage_cuvintele_din_linia_de_intrare;
  lmax:=0;
  for i:=1 to n do
    if Length(cuv[i]) > lmax then lmax:=Length(cuv[i]);
  Writeln('Cuvintele cu lungimea maxima=',lmax,' sunt:');
  for i:=1 to n do
    if Length(cuv[i])=lmax then Write(cuv[i],' ');
  Writeln
End.
```

57. *Vârsta medie*

Se dau numele şi vârsta a n copii pe n linii de intrare. Numele şi vârsta sunt separate prin spaţii. Se cere să se afişeze numele copiilor cu vârsta cea mai apropiată de media vârstei lor.

Soluţie:

Problema revine la calculul mediei vârstei şi apoi a minimului diferenţelor (în valoare absolută) dintre această medie şi vârsta fiecărui copil. Pentru aceasta numele şi vârstele trebuie extrase din liniile de intrare şi introduse în două tablouri.

```
Program Copiii_cu_varsta_cea_mai_apropiata_de_varsta_medie;
var nume:array[1..20] of string;
    v:array[1..20] of Byte;
    linie:string;
    i,n:Byte;
    med,min,dif:Real;

procedure Extragere_nume_si_varsta_din_liniile_de_intrare;
var i:Byte;
    c:Char;
begin
  for i:=1 to n do
  begin
    Read(c);
    while c=' ' do Read(c);   { citim ' ' de la inceputul liniei }
    nume[i]:='';
    while c<>' ' do                  { citim numele copilului "i" }
    begin nume[i]:=nume[i]+c; Read(c) end;
    Readln(v[i])                     { citim varsta copilului "i" }
  end
end;

Begin
  Write('n='); Readln(n);
```

```
  Writeln('Dati numele si varsta:');
  Extragere_nume_si_varsta_din_liniile_de_intrare;
  med:=0;
  for i:=1 to n do med:=med+v[i];
  med:=med/n;
  min:=200;
  for i:=1 to n do
  begin
    dif:=Abs(med-v[i]);
    if dif < min then min:=dif
  end;
  Write('Copiii_cu_varsta_cea_mai_apropiata_de_medie=');
  Writeln(med:5:2,' sunt:');
  for i:=1 to n do
  begin
    dif:=Abs(med-v[i]);
    if dif = min then Writeln( nume[i],' ',v[i])
  end
End.
```

58. Carte

O carte pentru copii are paginile împărţite în 3 părţi, fiecare parte putând fi răsfoită separat. Fiecare din cele 3 părţi ale unei pagini cuprinde una din cele trei silabe ale unui nume de animal tipărit pe ambele feţe ale paginii (de exemplu: cro–co–dil, e–le–fant etc.). Răsfoirea separată a celor 3 părţi ale cărţii produce, spre deliciul copiilor, nume de animale imaginare (de exemplu: cro–li–fant, e–co–dil etc.). Să se afişeze toate cuvintele care se pot citi prin această răsfoire ciudată a cărţii. Se introduc: numărul de pagini ale cărţii şi cuvintele despărţite în silabe de pe fiecare pagină. Datele de intrare se presupun corecte.

Soluţie:

Cartea se poate codifica printr-o matrice pag_{3xn} care cuprinde silabele cuvântului fără linioarele de despărţire în silabe pentru a afişa cuvântul.

Programul utilizează funcţia `Pos` pentru a depista poziţia primei "–", preia cu funcţia `Copy` prima silabă şi o memorează în matrice, iar apoi o şterge din *string*-ul citit din linia de intrare. Procedează în continuare la fel şi pentru restul silabelor. (Pentru aceasta a fost nevoie să se adauge o "–" la sfârşitul cuvântului.)

```
Program Jocul_silabelor;
var pag:array[1..3,1..30] of string;
    cuv:string; i,j,k,n,lg_silabei:Byte;
Begin
  Write('nr.pagini='); Readln(n);
```

Scanned with OKEN Scanner

```
  for j:=1 to n do
  begin
    Write('Cuvantul despartit in cele 3 silabe:');
    Readln(cuv); cuv:=cuv+'-';
    for i:=1 to 3 do
    begin
      lg_silabei:=pos('-',cuv)-1;
      pag[i,j]:=Copy(cuv,1,lg_silabei);
      Delete(cuv,1,lg_silabei+1)
    end
  end;
  Writeln('Cuvintele care se pot forma sunt:');
  for i:=1 to n do
  begin
    for j:=1 to n do
      for k:=1 to n do
        Write(pag[1,i]+pag[2,j]+pag[3,k]+' ');
      Writeln
  end
End.
```

59. *Caractere* 'AB'

Fie un şir de 2n caractere cuprinzând două steluţe alăturate, n-1 caractere egale cu A şi n-1 caractere egale cu B. Se afişează toate mutările prin care cele două steluţe alăturate se înlocuiesc cu două caractere succesive din şir, în scopul aducerii şirului la forma cu toate A-urile la început şi toate B-urile la sfârşit, fără să conteze poziţia steluţelor. Dacă problema nu are soluţie, se afişează mesajul "Imposibil".

Soluţie:

Problema nu are soluţie decât dacă valorile lui n sunt mici, deoarece, în aceste cazuri, posibilităţile de mutare sunt mult mai reduse.

Exemple:

```
BA**BA          Imposibil

BAB**ABA        Stelutele se muta din 4 in 1
                Stelutele se muta din 1 in 5
                Stelutele se muta din 5 in 3
                Stelutele se muta din 3 in 7
                Stelutele se muta din 7 in 5
                Stelutele se muta din 5 in 2
```

Algoritmul încearcă să mute caracterele A pe primele n-1 poziţii din şir; la fiecare pas se transferă un A, doi A sau eventual nici unul (dacă poziţia respectivă este deja ocupată de A). Pentru a aduce un A pe o anumită poziţie, se caută în dreapta acesteia

primul A. Dacă poziţia sa nu coincide cu locul pe care trebuie adus atunci, pentru a fi posibil transferul, vor trebui aduse întâi steluţele în acel loc (dacă ele nu se găsesc deja acolo). Aducerea steluţelor pe poziţia respectivă se poate face printr-o singură mutare - dacă distanţa dintre această poziţie şi aceea a steluţelor este mai mare decât 1, sau prin două mutări: mutarea lor cu două poziţii la dreapta şi apoi cu trei poziţii la stânga, în caz contrar.

Mutarea steluţelor este realizată în procedura Muta; determinarea poziţiei steluţelor şi a primului A care trebuie adus se face cu funcţia Pos.

Şirul poz cuprinde poziţiile succesive ale steluţelor; cu ajutorul lui se face afişarea mutărilor, dacă problema are soluţie. Evident, dacă un A nu se poate aduce pe locul său, problema nu are soluţie şi programul se opreşte forţat. Acest caz este testat în procedura Muta.

Programul cuprinde şi funcţia booleană Sir_ordonat care verifică dacă şirul citit satisface cerinţele problemei, caz în care nu mai trebuie făcută nici o mutare. Această verificare este necesară, deoarece algoritmul aduce A-urile pe primele n-1 poziţii, dar s-ar putea ca A-urile să fie la începutul şirului, fără să ocupe primele n-1 poziţii.

```
Program Sir_A_B;
var a:string[50];                          { a este sirul initial }
    poz:array[0..50] of Byte;
    n,n2,i,k,p:Byte;
function Sir_ordonat:Boolean;
var b:Boolean;       { sir neordonat daca dupa un B apare un A }
begin
  Sir_ordonat:=true; b:=false;             { N-a aparut nici un B }
  for i:=1 to n2 do
    case a[i] of
      'A':if b then begin Sir_ordonat:=false; i:=n2 end;
      'B':b:=true
    end
end;
procedure Muta(i,j:Byte);    { Se muta stelutele din pozitia i }
begin                                              { in pozitia j }
  if (Abs(i-j)>1) and (j in [1..n2-1])
                          {necesara in apelurile artificiului }
  then
    begin
      if k=0 then poz[k]:=i;
      k:=k+1; poz[k]:=j;
      a[i]:=a[j]; a[i+1]:=a[j+1];
      a[j]:='*'; a[j+1]:='*'
    end
  else begin Writeln('Imposibil'); Halt end
end;
```

```
Begin
  Writeln('Dati sirul cu n-1 A-uri, n-1 B-uri ');
  Writeln('    si 2 stelute consecutive:');
  Readln(a);
  n2:=Length(a);
  n:=n2 div 2;
  if Sir_ordonat
  then Write('Sir satisface conditiile problemei - 0 mutari')
  else
    begin
      k:=0;                                  { k=indice in sirul poz }
      for p:=1 to n-1 do    { p=pozitia pe care trebuie adus A }
        if a[p]<>'A'
        then
          begin
            i:=Pos('*',Copy(a,p,50))+p-1;
            if i-p=1
            then
              begin
                Muta(i,i+2);
                Muta(i+2,p)
              end
            else
              if i<>p then Muta(i,p);
                                  { stelutele sunt pe pozitia p }
            i:=Pos('A',Copy(a,p+1,50))+p;
            if i<n2 then Muta(p,i)
                    else
                      begin
                        Muta(p,n2-1);                  { p=n-1 }
                        Muta(n2-1,n+1);
                        Muta(n+1,n-2)
                      end
          end;
      for i:=0 to k-1 do
      begin
        Write('Stelutele se muta din ',poz[i]);
        Writeln(' in ',poz[i+1])
      end
    end
End.
```

Scanned with OKEN Scanner

II. SISTEME DE GESTIUNE A BAZELOR DE DATE

1. Repartizare în clase

În directorul C:\ATESTAT în fişierul **ELEVI.DBF** sunt trecuţi elevii admişi în clasa a IX-a.

ELEVI.DBF:

NUME	PREN	MEDIE	SEX
C,15	C,15	N,5,2	C,1

Numărul de clase (n>=2 – divizor al numărului de înregistrări din fişier) în care urmează să fie repartizaţi aceşti elevi se citeşte de la tastatură.

Să se creeze prin program n fişiere, denumite 9A,9B,..., corespunzătoare celor n clase în care vor fi repartizaţi elevii reuşiţi la admitere, respectând următoarele cerinţe:
a) numărul fetelor să fie egal cu numărul băieţilor în fiecare clasă; dacă acest lucru nu este posibil, diferenţa dintre numărul de fete şi numărul de băieţi să fie minimă;
b) repartizarea trebuie să fie relativ echilibrată şi din punct de vedere al mediilor.

Exemplu:

n=3

9A	9B	9C
1	2	3
6	5	4
7	8	9
	...	10

unde 1,2,... reprezintă elevii în ordinea descrescătoare a mediilor

Soluţie:

Pentru rezolvarea problemei se poate proceda astfel:
– ordonăm fişierul **ELEVI.DBF** după sex şi medii (transformăm media în şir de caractere);
– creăm automat structura fişierelor claselor a 9-a folosind funcţia sistem CHR() şi macrosubstituţia;
– parcurgem fişierul **ELEVI.DBF** şi introducem elevii în fişierele claselor a 9-a create anterior după modelul din enunţ. Fişierul în care introducem elevul curent este selectat ca în pasul anterior (cu funcţia CHR() şi macrosubstituţie) iar direcţia se stabileşte cu ajutorul unei variabile mers care ia valoarea 1 când mergem spre dreapta şi 2 când mergem invers.

```
set talk off
set safe off
close data
clea
use elevi
inde on sex+str(medie,5,2) tag smed &&reusitii dupa sexe si medii
```

Scanned with OKEN Scanner

```
inpu 'Cate clase sunt: ' to nrc
for i=1 to nrc        &&se creeaza structura fisierelor 9A,9B,...
  n=chr(64+i)
  copy stru to 9&n
endf
mers=1
i=1
do while !eof('elevi')
  sele b
  cl=chr(64+i)
  use  9&cl
  appe blan
  repl nume with elevi.nume,medie with elevi.medie
  repl sex with elevi.sex
  i=iif(mers=1,i+1,i-1)
  do case
    case i>nrc
      i=nrc
      mers=2
    case i=0
      i=1
      mers=1
    endc
  skip in elevi
endd
retu
```

2. Concurs de orientare turistică

În directorul `C:\ATESTAT` există fişierele **CONCURS.DBF** şi **START.DBF**

CONCURS.DBF:

NR_CAT	TSTART
N,5	C,5

unde:

– NR_CAT este un cod zecimal (primele 2 cifre reprezintă categoria, iar următoarele 3 cifre, numărul de concurs al persoanei în cadrul categoriei);

– TSTART este un câmp în care se va scrie ora la care va porni concurentul având numărul de concurs reprezentat prin ultimele 3 cifre din câmpul NR_CAT. Câmpul TSTART are formatul hh.mm.

START.DBF:

CAT	TIMPS	DELTA
N,2	N,4	N,2

unde:

– CAT este categoria de concurs;

– TIMPS este ora de plecare a categoriei respective în formatul hhmm);
– DELTA este intervalul de timp (în minute) dintre concurenții categoriei respective.

a) Se cere să se completeze câmpul TSTART din fișierul **CONCURS.DBF** pe baza datelor existente în fișierul **START.DBF**
b) Să se afișeze pe ecran ordinea cronologică în care vor porni concurenții.

Exemplu:

```
Timp de plecare    Numere de concurs
10.00              301
10.10              302
10.20              303 401
```

Soluție:

Ideea de rezolvare este firească:
– ordonăm fișierul concurs după câmpul NR_CAT;
– parcurgem fișierul pe categorii și completăm câmpul TSTART, calculând valoarea acestuia în funcția utilizator ts()

```
set talk off
set safe off
close data
clea
use concurs
inde on nr_cat tag cat
sele b
use start
inde on cat tag cate
sele a
do while !eof()
  c=int(nr_cat/1000)
  nr=1
  do while c=int(nr_cat/1000)
    repl tstart with ts(c,nr)
    nr=nr+1
    skip
  endd
endd
inde on tstart tag ts
disp all tstart,nr_cat
set talk on
set safe on
retu
```

Scanned with OKEN Scanner

```
function ts
  para c,nr
  sele b
  if !seek(c)
    ?'Date inconsistente!'
    wait wind 'Corectati si reluati executia!'
    sele a
    retu 'Gresa'
  else
    ora=int(timps/100)
    min=(timps-ora*100)+delta*(nr-1)
    do while min>=60
      min=min-60
      ora=ora+1
    endd
    sele a
    retu str(ora,2)+'.'+iif(min>0,str(min,2),str(min,1)-'0')
  endi
```

3. Mişcările pe beneficiari

În directorul `C:\ATESTAT` există fişierele **`TRANZAC.DBF`** şi **`BENEF.DBF`** având următoarele structuri:

`TRANZAC.DBF`:

CODB	DATA	TIPTRAN	DEBIT	CREDIT	CANT
N,3	D	C,4	N,8	N,8	N,10

unde:

– `CODB` este codul beneficiarului;
– `DATA` este data tranzacţiei (livrare sau plata);
– `TIPTRAN` este tipul tranzacţiei (`F` – factură, `DP` – dispoziţie de plată);
– `DEBIT` este valoarea din factură;
– `CREDIT` este valoarea la plată (dispoziţie de plată);
– `CANT` este cantitatea livrată în cazul `TIPTRAN='F'`, şi 0 în cazul `TIPTRAN='DP'`;

`BENEF.DBF`:

CODB	NUMEB	ADRB	SOLD
N,3	C,25	C,20	N,10

unde:

– `CODB` este codul beneficiarului;
– `NUMEB`, `ADRB` sunt numele şi adresa beneficiarului;
– `SOLD` este valoarea cu care este dator (`DEBIT–CREDIT`).

Presupunem că ne aflăm în ultima zi a lunii curente şi trebuie listate mişcările pe beneficiari, adică:

Scanned with OKEN Scanner

a) se cere un tabel având următorul antet:

NUME BENEFICIAR	SOLD INITIAL	LIVRARI	PLATI	SOLD FINAL

b) La sfârşitul listei trebuie să apară o linie cu totaluri.

Observaţie:

– SOLD_INITIAL înseamnă soldul la începutul lunii;
– SOLD_FINAL înseamnă soldul la sfârşitul lunii;
– Fişierele **TRANZAC.DBF** şi **BENEF.DBF** există şi conţin cel puţin 10 articole;
– Se consideră că aceste fişiere sunt corect create, nu trebuie făcute validări.

Soluţie:

– vom extrage în fişierul **LUNCU.DBF** (luna curentă) articolele corespunzătoare datelor calendaristice citite;
– din acest fişier, cu ajutorul comenzii TOTAL vom crea fişierele **LIV.DBF** şi **BANCA.DBF** cuprinzând total livrări, respectiv total plăţi pentru fiecare beneficiar;
– realizăm o legătură relaţională între fişierul **BENEF.BDF** (tată) şi fişierele **LIV.DBF** şi **BANCA.DBF** (fii) prin intermediul câmpului comun codb şi calculăm soldul iniţial pentru fiecare beneficiar si=BENEF.sold+BANCA.credit-LIV.debit.

Evident vom prelucra numai beneficiarii care au avut "mişcări" în luna curentă.

```
set talk off
set safe off
close all
set date to brit
use tranzac
inpu 'INTR. PRIMA ZI A LUNII CURENTE-{ZZ/LL/AA}-:' to d1
inpu 'INTR. ULTIMA ZI A LUNII CURENTE-{ZZ/LL/AA}-:' to d2
copy to luncu for (data>=D1).and.(Data<=D2)
use luncu
inde on codb tag cod
set orde to cod
total on codb to liv fiel debit for tiptran='F'
use liv in b
sele b
inde on codb tag cod
sele a
tota on codb to banca fiel credit for tiptran='DP'
use banca in c
sele c
inde on codb tag cod
sele a
use benef   orde numeb
acce 'INTR. LUNA:' to lun
```

Scanned with OKEN Scanner

```
set rela  to codb into b, codb into c
nr=0
totsi=0
?'                                        SITUATIA MISCARILOR'
?'                                    PE BENEFICIARI LUNA',LUN
?'_____________________________________________________________________'
?'|Nr.|  Denumire   | Sold     | Valoare la| Platit in | Sold  |'
?'|crt| Beneficiar | initial  | pr.vin.   | Banca     | final |'
?'|___|____________|__________|___________|___________|_______|'
scan
  if sold<>0 or b.debit<>0 or c.credit<>0
    si=sold+c.credit-b.debit
    nr=nr+1
    ?'|'+str(nr,2)+'|'+numeb+'|'+str(si,10)+'|'+str(b.debit,8)+;
    '|' +str(c.credit,8)+'|'+str(sold,10)+'|'
    totsi=totsi+si
  endi
ends
?'|-----------------------------------------------------------|'
sum b.debit,c.credit,sold to tpl,tpba,totsf
?' '
?' TOTAL :      '+str(totSI,11)+' '+STR(TPL,9)+' '+STR(TPBA,9)+;
                                                ' '+STR(TOTSF,9)

close all
set safe on
set talk on
retu
```

4. La maternitate

În directorul `C:\ATESTAT` există fişierele **`LOCALITATE.DBF`**, **`BEBE.DBF`**, **`PARINTI.DBF`** şi **`SPITAL.DBF`** având următoarele structuri:

`LOCALITATE.DBF`:

CODS	LOCALITATE
N,2	C,15

unde:

– `CODS` este codul spitalului;

`BEBE.DBF`:

NUME	PRENUME	DATA	CODS	COD
C,15	C,15	D	N,2	N,3

unde:

– `COD` este codul copilului;

– `DATA` este data naşterii;

PARINTI.DBF:

NUME	PRENUME	PRENMA	COD
C,15	C,15	C,15	N,3

unde: – NUME şi PRENUME sunt numele şi prenumele tatălui;
– PRENMA este prenumele mamei;

SPITAL.DBF:

CODS	DENUS
N,2	C,20

unde: – DENS este numele spitalului.

a) Să se afişeze toţi copiii născuţi în luna x şi anul y (citite de la tastatură) din toate localităţile astfel:

nume	initiala_tatalui	prenume	[G]

– eventualii gemeni vor apare unul după celălalt marcaţi cu G;

b) Să se afişeze o situaţie statistică cu numărul de copii născuţi pe o perioadă de timp dată:

nume	spital	localitate	Nr. copii

c) Care este rolul câmpului PRENMA?

Soluţie:

a)

– realizăm o legătură relaţională a fişierului **BEBE.DBF** cu fişierul **LOCALITATE.DBF** (prin intermediul câmpului cods) şi cu fişierul **PARINTI.DBF** (prin câmpul cod);
– parcurgem fişierul **BEBE.DBF**, ordonat după codul spitalului (câmpul scod) şi filtrat după luna şi anul, citite de la tastatură. Parcurgerea se efectuează în două structuri repetitive (DO WHILE) imbricate; prima în cadrul aceluiaşi spital şi a doua pentru eventualii gemeni (coincidenţa datei de naştere, a numelui de familie, a prenumelui mamei (răspuns şi la punctul c)).

b)

– realizăm o legătură relaţională a fişierului **BEBE.DBF** cu fişierul **LOCALITATE.DBF** şi cu fişierul **SPITAL.DBF** (prin intermediul câmpului cods);
– parcurgem fişierul **BEBE.DBF**, ordonat după codul spitalului (câmpul cods) şi filtrat după perioada citită de la tastatură.

```
set talk off
set date to brit
close all
clea
inpu 'Luna:' to lun
inpu ' Anul:' to an
use local in b orde cods
use bebe
use parinti in c orde cod
use spital in d orde cods
```

```
set orde to codsd
set rela to cods into b, cod into c
set filt to month(data)=lun and year(data)=an
go top
do while !eof()
  spi=cods
  ?'In localitatea:',b.local,'s-au nascut:'
  do while cods=spi
    dn=data
    nb=nume
    pt=c.prenume
    pm=c.prema
    g=0
    do while dn=data and nb=nume and pt=c.prenume and pm=c.prema
      if g=0
        ?nume,subs(ltri(c.prenume),1,1)+'.',prenume
      else
        @ ROW(),COL()+1 SAY iif(g=1,'G','')
        ?nume,subs(ltri(c.prenume),1,1)+'.',prenume+' G'
      endi
      g=g+1
      skip
    endd
  endd
endd
*Punctul b*
inpu 'Inceput perioada {zz/ll/aa}:' to d1
inpu 'Sfarsit perioada {zz/ll/aa}:' to d2
set filter to data>=d1 and data<=d2
go top
set orde to cods
set rela to cods into d, cods into b
do while !eof()
  spi=cods
  ns=d.dens
  loc=b.local
  nrc=0
  do while cods=spi
     nrc=nrc+1
     skip
  endd
  ?ns,spi,loc,nrc
endd
close all
set talk on
retu
```

Scanned with OKEN Scanner

5. *Orar*

În directorul C:\ATESTAT există fişierele **ORE.DBF** şi **DISCIPLINE.DBF**:

ORE.DBF:

```
CLASA   ZI   ORE
 C,3    N,1  C,7
```

unde:

– ZI reprezintă numărul zilei din săptămână (luni=1, marţi=2,...);

– ORE sunt disciplinele din orar în ziua respectivă (în ordinea din orar); fiecare disciplină fiind codificată pe un caracter.

DISCIPLINE.DBF:

```
ORE DEN_DISCIP
C,1   C,10
```

unde DEN_DISCIP este denumirea în clar a materiei.

Să se listeze la cerere orarul unei clase al cărei cod este în câmpul ORE conform următoarei structuri:

```
    Luni     Marti     ...
8   Romana   Franceza
9   Fizica   Chimie
10  ...      ...
```

Soluţie:

– datorită modului în care avem codificat orarul în cele două fişiere vom folosi pentru a extrage şi lista orarul unei clase, o matrice ora[7,6], cu 7 linii pentru că o clasă poate avea maximum 7 ore într-o zi şi 6 coloane pentru că o coloană o vom folosi pentru specificarea orei de începere a lecţiei.

– în prelucrarea unui articol din acest fişier, (**ORE.DBF**) vom extrage din valoarea câmpului ORE câte un caracter corespunzător unei materii pe care îl vom căuta în fişierul **DISCIPLINE.DBF** şi vom memora în matricea ora denumirea în clar a materiei. Atragem atenţia la cazurile întâlnite cu ocazia acestor căutări: codul materiei din fişierul **ORE.DBF** nu există în fişierul **DISCIPLINE.DBF** dintr-o eroare la încărcarea datelor sau nu există pentru că acel caracter este blanc şi deci clasa nu mai are ore în ziua respectivă sau nu există pentru că acel caracter este blanc dar în continuare există caractere diferite de blanc respectiv clasa are "oră liberă" (fereastră) ceea ce înseamnă că, din nou, este vorba de o eroare la încărcarea datelor. Aceste cazuri le tratează procedura veri, lucru neimpus explicit de enunţ.

```
set talk off
clea
close all
dime ora(7,6)
```

Scanned with OKEN Scanner

```
for i=1 to 7
  for j=1 to 6
    ora(i,j)=iif(j=1,str(i+7,2)+' ' ,' ')
  endf
endf
use ore orde cls
use disci in b
acce  ' Introduceti clasa:' to cl
if !seek(cl)
  ?'Clasa introdusa nu exista!'
else
  do while clasa=cl
    j=zi+1 && prima coloana a fost completata!
    for i=1 to 7
      ma=subs(ore,i,1) && codul materiei "curente"
      sele b
      loca for ore=ma
      bun=.t.
      do veri with bun
      if !bun
        retu
      endi
      ora(i,j)=den_d
      sele a
    endf
    skip
  endd
  clea
  ?'Orarul pentru clasa:'+cl
  ?''
  ?'  luni   ','  marti  ',' miercuri  ', ' joi   ', ' vineri '
  ?''
  for i=1 to 7
    for j=1 to 6
      ??ora(i,j)
    endf
    ?''
  endf
endi
set talk on
retu
```

Scanned with OKEN Scanner

```
proc veri
para bun
if! found()
  if ! found() and ma<>' '
    ?'INCONSISTENTA, MATERIE NEDEFINITA:'+MA
    wait nowait 'Verificati datele!'
    bun=.f.
  else
    if !empty(subs(a.ore,i+1))
      ?'Ferestra la clasa!'
      wait nowait '   Verificati datele!'
      bun=.f.
   endi
  endi
endi
retu
```

6. *Condensare de fişiere*

Se introduce de la tastatură (validare) numele unui fişier de date din directorul `C:\ATESTAT`. Să se condenseze fişierul citit, prin condensare înţelegându-se modificarea lungimii câmpurilor de tip caracter şi/sau numeric la dimensiunea minimă, astfel încât să nu se trunchieze valorile câmpurilor. Fişierul rezultat va fi redenumit cu numele fişierului citit.

Soluţie:

Alegem fişierul dorit şi vom efectua asupra lui următoarele operaţii:
– eliminăm spaţiile din stânga şi dreapta valorilor atributelor de tip şir de caractere, respectiv zerourile nesemnificative în valorile atributelor de tip numeric, aflăm lungimea maximă utilă (în funcţie de valorile concrete conţinute în fişier) şi modificăm structura cu această valoare;
– creăm (automat) noua structură;
– copiem articolele din vechiul fişier, ştergem fişierul vechi şi redenumim fişierul nou cu vechiul nume. Evident, fişierul obţinut rămâne utilizabil în orice aplicaţie în care este folosit ca fişier `.dbf`.

```
set talk off
set safe off
close all
clea
defi wind w1 from 5,5 to 22,75 titl 'Program de "condensare"'
defi menu men in wind w1
defi pad p1 of men promp 'Alege';
             mess'Permite alegerea fisierului pentru condesare'
```

Scanned with OKEN Scanner

```
defi pad p2 of men promp 'Iesire';
                                mess'Permite iesirea din program'
defi popu pp1 from 1,4 prom file like *.dbf shad
defi popu pp2 from 1,10 shad
defi bar 1 of pp2 prom 'In mediu '
defi bar 2 of pp2 prom 'In DOS'
on sele  pad p1 of men acti popu pp1
on sele popu pp1 do proc1
on sele  pad p2 of men acti popu pp2
on sele bar 1 of pp2 do ies
on sele bar 2 of pp2 quit
acti wind w1
acti menu men
retu
**********
proc ies
  close all
  deact wind w1
  deac menu
  set talk on
  set safe on
retu
**********
proc proc1
  f=prom()
     && f contine numele fisierului selectat pentru "condensare"
  use &f
  copy stru exte to fstr
  use fstr in b
  sele b
  scan for field_type='C'or field_type='N'
    stor 0 to lc,ln,ld
    if field_type='C'
      do con1 with lc
      repl field_len with lc;
                 && Modifica lungimea cimpului caracter corespun-
    else                 && zator celei mai lungi valori din fisier
      do con2 with ln,ld
      repl field_len with ln, field_dec with ld;
   && Modifica lungimea cimpului numeric corespunzator celei mai
                                        && mari valori din fişier
    endi
  ends
  close all
  crea fnou from fstr
  appe from &f
```

```
  eras &f
  close all
  rena fnou.dbf to &f
  wait wind 'Operatia s-a incheiat (cu succes)! Apasati o tasta.'
  deact popu pp1
retu
**********
proc con1 && Returneaza lungimea cea mai mare a valorii cimpului
para max  && de tip caracter curent
  sele a
  g=b.field_name
  max=len(alltrim(&g))
  scan
    if len(alltrim(&g))>max
      max=len(alltrim(&g))
    endi
  ends
  sele b
retu
**********
proc con2 && Returneaza lungimea cea mai mare a valorii campului
para max1,max2                          && de tip numeric curent
  sele a
  g=b.field_name            && g contine numele campului numeric
  max1=len(alltrim(str(&g)))
  nrz=str(&g,b.field_len,b.field_dec)
  poz=at('.',nrz)+1
  max2=len(zero(substr(nrz,poz)))
  scan
    nrz=str(&g,b.field_len,b.field_dec)
    poz=at('.',nrz)+1
    if len(alltrim(str(&g)))>max1
      max1=len(alltrim(str(&g)))
    endi
    if len(zero(substr(nrz,poz)))>max2
      max2=len(zero(substr(nrz,poz)))
    endi
  ends
  if b.field_dec<>0
    max1=max1+max2+1
  endi
  sele b
retu
```

```
func zero &&Functie pentru eliminarea zerourilor nesemnificative
para nr                                  && din partea fractionara
  nrc=val(nr)
  set deci to b.field_dec
  elim=0
  for i=1 to b.field_dec
    if nrc/10**i=int(nrc/10**i)
      elim=elim+1
    endi
  endf
  set deci to
retu subs(nr,1,b.field_dec-elim)
```

7. *Operații pe fișiere*

Să se scrie un program care realizează întreținerea unui fișier de date. Se va crea un meniu de forma:

Fisier	Operatii	Iesire

Pentru opțiunea "Fisier" se va defini un popup cu numele fișierelor, iar pentru opțiunea "Operatii" un popup cu operațiile: adăugări, modificări și ștergeri de articole.

Soluție:

```
set talk off
set safe off
close all
clea
publ nc,nuc &&nume camp, nume camp curent
defi wind w1 from 5,5 to 22,75 titl 'Program de intretinere' doub
defi menu men in wind w1
defi pad p1 of men promp 'Alege ';
                                  mess'Permite alegerea fisierului '
defi popu pp1 from 1,4 prom file like *.dbf shad
on sele  pad p1 of men acti popu pp1
on sele popu pp1 do proc1
acti wind w1
acti menu men
retu
**********
proc proc1
  hide menu men
  hide popu pp1
  f=prom()
       &&f contine numele fisierului selectat pentru intretinere
  use &f
  vedems=iif(recc()=0,.t., .f.)
```

```
  copy stru exte to fstr
  use fstr in b
  sele b
  nc=recc()
  NUC=field_name
  defi menu men2
  defi pad p3 of men2 promp 'Adaugari';
                                mess'Permite adaugari de artiole'
  defi pad p4 of men2 promp 'Modificari';
           mess'Permite modificari de; articole'skip for vedems
  defi pad p5 of men2 promp 'Stergeri';
              mess'Permite stergeri de; artcole'skip for vedems
  defi pad p2 of men2 promp 'Iesire';
                                mess'Permite iesirea din program'
  defi popu pp3 from 1,19 prom fiel b.field_name;
           mess 'Alegeti campul; dupa care selectati articolul'
  defi popu pp2 from 1,44 shad
  defi bar 1 of pp2 prom 'In mediu '
  defi bar 2 of pp2 prom 'In DOS'
  on sele  pad p2 of men2 acti popu pp2
  on sele  pad p3 of men2 do ad
  on sele  pad p4 of men2 acti popu pp3
  on sele popu pp3  do mo
  on sele  pad p5 of men2 acti popu pp3
  on sele bar 1 of pp2 do ies
  on sele bar 2 of pp2 quit
  acti menu men2
retu
**********
proc ies
  close all
  deact wind w1
  deac menu all
  set talk on
  set safe on
retu
**********
proc ad
  hide menu men
  deact wind w1
  clea
  sele a
  a='D'
  do while uppe(a)='D'
    appe blan
```

Scanned with OKEN Scanner

```
    for i=1 to nc
      vg=field(i)
      @i,3 say 'Campul '+fiel(i) get &vg
    endf
    @i+2,5 say 'Mai adaugati <d/n>:' get a
    read
  endd
  clea
  acti wind w1
  show menu men
retu
**********
proc mo
  NUC=prom()
  sele a
  defi popu ppm from 1,25 prom fiel &nuc;
                mess 'Alegeti articolul pt. modificare/stergere'
  on sele popu ppm do mop
  acti  popu ppm
retu
**********
proc mop
  hide menu men
  hide popu ppm, pp3
  deact wind w1
  clea
  if pad()='P4'&&modificare sau stergere
    a='D'
    for i=1 to nc
      vg=field(i)
      @i,3 say 'Campul '+fiel(i) get &vg
    endf
    @i+2,5 say 'Mai modificati <d/n>:' get a
    read
    if uppe(a)='D'
      clea
      acti wind w1
      show menu men
    else
      clea
      acti wind w1
      show menu men
      deacti popu ppm
    endi
```

Scanned with OKEN Scanner

```
    else
      a='D'
      disp
      @row()+2,5 say 'Acesta este articolul care va fi sters?;
                                                    '<d/n>:' get a
      read
      if uppe(a)='D'
        dele
        pack
      endi
      @ row()+1,5 say 'Mai doriti sa stergeti? <d/n>:' get a
      read
      if uppe(a)='D'
        clea
        acti wind w1
        show menu men
      else
        clea
        acti wind w1
        show menu men
        deacti popu ppm
      endi
    endi
 endi
retu
```

8. Evidența profesorilor

Se dau fișierele **ELEVI.DBF**, **DIRI.DBF** și **PROFI.DBF**, având următoarele structuri:

ELEVI.DBF:

```
NUME  CLS
C,20  C,3
```

unde CLS este clasa;

DIRI.DBF:

```
CLS  NUME
N,3  C,20
```

PROFI.DBF:

```
NUME CLS1  CLS2 ... CLS18   MAT
C,20 C,3   C,3  ...  C,3    C,15
```

unde MAT este materia predată la clasa CLS1, CLS2,...

Observație:

Dacă un profesor nu predă la 18 clase atunci câmpurile rămase libere vor conține blancuri.

Scanned with OKEN Scanner

Să se realizeze un program care afişează pe ecran:
a) profesorii unei clase, ştiind că o clasă se identifică prin numele ei sau prin numele dirigintelui (introdus fie unul, fie celălalt de la tastatură);
b) profesorii care predau o anumită disciplină (introdusă de la tastatură); se vor afişa numele profesorului, clasele la care acesta predă, precum şi numele dirigintelui fiecărei clase afişate;
c) profesorii diriginţi care nu au ore cu clasa lor în afara orei de dirigenţie.

Soluţie:

a)

Pentru căutarea profesorilor unei clase am realizat procedura `pro`, în care am scanat fişierul **`PROFI.DBF`**, apoi cu ajutorul funcţiei standard `field()` şi a macrosubstituţiei am decis dacă profesorul "curent" predă sau nu la clasa respectivă (cea citită sau indicată prin diriginte). Operaţia de căutare, se realizează într-un `for` controlat de numărul maxim de clase la care un profesor poate avea ore (18 clase).

b) şi c)

Folosim procedee asemănătoare, fără să apelăm funcţia `pro`.

```
set talk off
clea
close data
sele 0
use elevi
sele 0
use diri
sele 0
use profi
mai='D'
?'Punctul a:'
do while uppe(mai)='D'
  clea
  acce 'Ce clasa doriti(numele clasei sau numele dirigintelui):';
                                                          to cln
  if val(subs(allt(uppe(cln)),1,1))#0
    sele diri
    loca for cls=cln
    if !found()
      ?'Clasa introdusa nu exista sau nu are diriginte!'
    else
      ?'Clasa:',cln,'are diriginte prof.:',nume
      ?'Si urmatorii profesori:'
      do pro with cln
    endi
```

```
  else
    sele diri
    loca for nume=cln
    if !found()
      ?'Dirigintele introdus nu exista sau nu are clasa!'
    else
      ?'Diriginte prof.:',nume,'are clasa:',cls
      ?'Si urmatorii profesori:'
      do pro with cls
    endi
  endi
  acce ' Mai doriti pentru alta clasa? <d/n>:' to mai
endd
clea
?'Punctul b'
acce 'Introduceti materia dorita:' to m
sele profi
loca for mat=m
if !found()
  ?'Materia introdusa nu exista!?'
else
  do while found()
    i=2
    cl=field(i)
    ?nume,'preda la clasele:'
    cc=!empty(&cl)
    do while cc
      sele diri
      loca for cls=profi->&cl
      sele profi
      ?&cl,'diriginte:',diri.nume
      i=i+1
      cl=field(i)
      cc=!empty(&cl)
    endd
    cont
  endd
endi
wait wind 'Pt. punctul c apasati o tasta'
clea
?'Punctul c (dirigintii care nu au ore la clasa lor)'
sele diri
scan
  sele profi
  loca for nume=diri.nume
  are=.f.
```

Scanned with OKEN Scanner

```
  for i=1 to 18
    cl=field(i+1)
    if &cl=diri.cls
      are=.t.
    endi
  endf
  sele diri
  if !are
    ?nume,'la clasa:',cls
  endi
ends
set talk on
close all
retu
**********
proc pro
para cln
  sele profi
  go top
  scan
    for i=1 to 18
      cl=field(i+1)
      if &cl=cln
        ? nume,'la:',mat
      endi
    endf
  ends
retu
```

9. *Fişiere cu structuri identice*

Să se creeze fişierul **FISIERE.DBF** având următoarea structură:

```
NUME
 C,8
```

Acest fişier conţine numele tuturor fişierelor cu extensia .DBF din directorul C:\ATESTAT.

Să se realizeze un program care listează pe ecran numele acelor fişiere care au structură identică. Dacă nu există asemenea fişiere, să se afişeze un mesaj corespunzător.

Soluţie:

– vom încărca fişierul de date **FISIERE.DBF** cu numele fişierelor de date existente pe unitatea de disc curentă cu ajutorul funcţiei ADIR();

– vom construi pentru primul fişier un fişier special care va conţine structura acestuia (comanda COPY STRUCTURE EXTENDED) şi o vom compara cu structurile celorlalte

Scanned with OKEN Scanner

fişiere din fişierul **FISIERE.DBF** (construind, cu aceeaşi comandă încă un fişier special de structură);
– dacă vom găsi structuri identice afişăm numele acestora şi ştergem numele fişierului duplicat din fişierul **FISIERE.DBF** (pentru a elimina simetria, fişierul F1 este identic structural cu F2 şi fişierul F2 cu F1). După epuizarea articolelor din fişierul **FISIERE** vom şterge primul articol din acest fişier (respectiv numele fişierului a cărui structură am comparat-o cu celelalte structuri);
– reluăm procesul cât timp în fişierul **FISIERE.DBF** mai sunt cel puţin două articole.
Observaţie:
Problema se reduce de fapt la compararea conţinutului a două fişiere de date.

```
set talk off
set safe off
publ num
clea
close database
erase fs1.dbf                    &&Nu prelucram fisierul de lucru!
erase fs2.dbf                    &&Nu prelucram fisierul de lucru!
=adir(fis,'*.dbf')
use fisiere
dele all
pack
appe from array fis
dele for nume='FISIERE'          &&Nu prelucram fisierul de lucru!
pack
repl all nume;
         with subs(nume,1,iif(at('.',nume)>0, at('.',nume)-1,8))
*Pastram numai numele fisierului fara extensie
copy to sfis
go top
do while recc()>1
  num=subs(nume,1)
  sele b
  use &num
  nfis=num
  copy stru exte to fs1
  use fs1
  inde on field_name tag nc
  do comp
  go top
  dele
  pack
endd
set talk on
set safe on
retu
```

Scanned with OKEN Scanner

```
proc comp
  sele a
  do while not eof()
    skip
    if not eof()
      num=subs(nume,1)
      go top in b
      sele c
      use &num
      copy stru exte to fs2
      use fsl
      inde on field_name tag nc
      if recc('b')=recc()
        do while not eof()
          bun=.t.
          if not((field_name=b.field_name);
         and(field_type=b.field_type)and(field_len=b.field_len);
                                        and(field_dec=b.field_dec))
            bun=.f.
            go bottom
          endi
          skip
          skip in b
        enddo
        if bun
          ?'Fisierul:'+nfis+'este identic cu:'+num
          sele a
          dele
        endi
      endi
      sele a
    endi
  enddo
retu
```

10. Operații monetare

Operațiile monetare realizate într-o bancă necesită o accesare simplă a următoarelor informații:
– evidența conturilor deschise;
– evidența operațiilor realizate în conturi (depunere/extragere din cont, creare/lichidare de cont).

Să se creeze o bază de date care modelează spațiul informațiilor și o aplicație (cu meniu) care să realizeze:

Scanned with OKEN Scanner

a) crearea fişierelor;
b) actualizarea sumelor curente din conturi (depuneri/extrageri);
c) determinarea numărului de operaţii realizate în fiecare cont;
d) determinarea sumei totale depuse/extrase în/din fiecare cont (zilnic, sau pe o perioadă dată).

Soluţie:

Se creează structurile următoarelor fişiere de date:

AOPER.DBF

NRC	DATA	DEBIT	CREDIT
N,4	D	N,8	N,8

unde – NRC este numărul contului, ce se va completa automat în program;
– DATA reprezintă data în care se creează contul;
– DEBIT reprezintă suma depusă;
– CREDIT reprezintă suma extrasă.

CONT.DBF

NRC	NUME	ADR	BI	SOLD	DATAI
N,4	C,20	C,30	C,8	N,10	D

unde – NUME, ADR, BI reprezintă datele persoanei;
– DATAI reprezintă data deschiderii contului;

OPER.DBF

NRC	DATA	DEBIT	CREDIT
N,4	D	N,8	N,8

```
set talk off
set date to brit
close all
clea
use aoper in c
use cont in b orde nrc
use oper
set rela to nrc into cont
defi menu m1
  defi pad contu of m1 prom 'CONTURI'
  defi pad ope of m1 prom 'OPERATII'
  defi pad stati of m1 prom 'STATISTICI' skip for eof('cont')
  defi pad ies of m1 prom 'IESIRE'
    defi popu pp1 from 1,1
       defi bar 1 of pp1 prom 'Selectare nr.cont'
       defi bar 2 of pp1 prom 'Selectare dupa nume'
       defi bar 3 of pp1 prom 'Cont nou'
```

Scanned with OKEN Scanner

```
      defi popu pp2 from 1,10
        defi bar 1 of pp2 prom 'Depunere'
        defi bar 2 of pp2 prom 'Extragere'
        defi bar 3 of pp2 prom 'Lichidare'
      defi popu pp3 from 1,15
        defi bar 1 of pp3 prom 'Nr. operatii cont'
        defi bar 2 of pp3 prom 'Dep./ext. cont pe zi'
        defi bar 3 of pp3 prom 'Dep./ext. cont pe o peri.'
      defi popu pp4 from 1,35
        defi bar 1 of pp4 prom 'In mediu'
        defi bar 2 of pp4 prom 'In DOS'
      defi popu pp5 from 2,1 prom fiel b.nrc mess' Alege un cont'
      defi popu pp6 from 3,1 prom fiel b.nume+b.bi mess'Alege un nume'
    on sele pad contu of m1 acti popu pp1
      on sele bar 1 of pp1 acti popu pp5
        on sele popu pp5 @ 0,43 say 'CONT SELECTAT:';
                                          +str(b.nrc)+space(20)
      on sele bar 2 of pp1 acti popu pp6
        on sele popu pp6 @ 0,43 say 'PERS. SELE.:';
                  +allt(b.nume)+' Cont:'+str(b.nrc,4)+space(10)
      on sele bar 3 of pp1 do cno
    on sele pad ope of m1 acti popu pp2
      on sele bar 1 of pp2 do dep
      on sele bar 2 of pp2 do ext
      on sele bar 3 of pp2 do lic
    on sele pad stati of m1 acti popu pp3
      on sele bar 1 of pp3 do nro
      on sele bar 2 of pp3 do dez
      on sele bar 3 of pp3 do depe
    on sele pad ies of m1 acti popu pp4
      on sele bar 1 of pp4 do ies
      on sele bar 2 of pp4 quit
acti menu m1
retu
**********
* procedura de creare a unui cont nou
proc cno
clea
sele cont
go bott
co=nrc+1
appe blan
@ 6,10 say 'Numele solicitantului:' get nume
@ 7,10 say 'Seria si nr. buletin:' get bi
@ 8,10 say 'Adresa:' get adr
@ 9,10 say 'Depozit initial:' get sold
```

Scanned with OKEN Scanner

```
@ 10,10 say 'Data deschiderii contului:'+dtoc(date())
@ 11,10 say 'Contul dvs. este:'+str(co)
read
@ 0,43 say 'CONT SELECTAT:'+str(b.nrc)
repl nrc with co,datai with date()
clea
retu
**********
* procedura de depunere in contul selectat
proc dep
@ 0,43 say 'CONT SELECTAT:'+str(b.nrc)
co=b.nrc
sele oper
conf='D'
appe blan
@ 6,20 say 'Data:' +dtoc(date())
@ 7,20 say 'Suma depusa:' get debit
@ 9,20 say 'Confirm<d/n>:' get conf
read
if uppe(conf)='D'
   repl nrc with co,data with date()
   sele  cont
   repl sold with sold+oper.debit
else
   dele
   pack
endi
sele oper
clea
retu
**********
* procedura de extragere a unei sume din contul selectat
proc ext
@ 0,43 say 'CONT SELECTAT:'+str(b.nrc)
co=b.nrc
so=b.sold
sele oper
conf='D'
appe blan
@ 6,20 say 'Data:' +dtoc(date())
@ 7,20 say 'Suma disponibila:'+str(so)
@ 8,20 say 'Suma extrasa:' get credit valid
iif(credit>so,.f.,.t.)
@ 9,20 say 'Confirm<d/n>:' get conf
read
```

Scanned with OKEN Scanner

```
if uppe(conf)='D'
  repl nrc with co,data with date()
  sele  cont
  repl sold with sold-oper.credit
else
  dele
  pack
endi
sele oper
clea
retu
**********
* procedura de lichidare a unui cont
proc lic
conf='D'
sele cont
@ 6,20 say 'Data:' +dtoc(date())
@ 7,20 say 'Suma existenta:'+str(sold)
@ 9,20 say 'Confirm<d/n>:' get conf
read
if uppe(conf)='D'
  dele
  pack
co=nrc
sele aoper
appe from oper for nrc=co
sele oper
dele for nrc=co
pack
endi
clea
retu
**********
* procedura de afisare a numarului de operatii realizate in
* contul selectat
proc nro
@ 0,43 say 'CONT SELECTAT:'+str(b.nrc)
sele aoper
nr=cont.nrc
nti=cont.nume
sum 1 to nro for nrc=nr
sum 1 to nrod for nrc=nr and debit>0
@ 6,30 say 'In contul selectat, titular: '+nti
@ 7,40 say 'S-au efecuat:'+str(nro+1)+' operatii'
@ 8,40 say 'Din care:'+str(nrod+1)+' depuneri'
wait wind 'Pt continuare apasati o tasta'
```

Scanned with OKEN Scanner

```
clea
retu
**********
* procedura de determinare a sumelor depuse/extrase
* in ziua curenta
proc dez
@ 0,43 say 'CONT SELECTAT:'+str(b.nrc)
nr=cont.nrc
nti=cont.nume
sum debit to sd for nrc=nr and data=date()
sum credit to se for nrc=nr and data=date()
@ 6,25 say 'In contul selectat, titular: '+nti
@ 7,40 say 'S-au depus:'+str(sd)+' lei'
@ 8,40 say 'S-au extras:'+str(se)+' lei'
wait wind 'Pt continuare apasati o tasta'
clea
retu
**********
* procedura de determinare a sumelor depuse/extrase
* intr-o perioada introdusa de la tastatura
proc depe
in={00/00/00}
fi={00/00/00}
@ 7,10 say 'Data intiala:' get in
@ 8,10 say 'Data finala:' get fi valid in<fi
read
nr=cont.nrc
nti=cont.nume
sum debit to sd for nrc=nr and (data>=in and data<=fi)
sum credit to se for nrc=nr and (data>=in and data<=fi)
@ 6,40 say 'In contul selectat, titular: '+nti
@ 7,40 say 'S-au depus:'+str(sd)+' lei'
@ 8,40 say 'S-au extras:'+str(se)+' lei'
wait wind 'Pt continuare apasati o tasta'
clea
retu
* procedura de iesire din program
* cu revenire in fereastra de comenzi
**********
proc ies
close all
clea
deact menu m1
retu
```

Scanned with OKEN Scanner

11. Rețete

Se dă un fișier de date **ALIMENTE.DBF** cu structura:

CODA	DENA	CALORII	PRET-UNIT
N,6	C,15	N,5	N,5

Să se scrie un program care să realizeze următoarele cerințe:

a) Construiește un fișier **RETETE.DBF** pe baza alegerii prin dialog a produselor alimentare care vor intra în structura fiecărei rețete.

Structura fișierului **RETETE.DBF** va fi următoarea:

NUMER	CODA1	CANTA1	CODA2	CANTA2	...	CODA5	CANTA5
C,20	N,6	N,4	N,6	N,4		N,6	N,4

TOTALCALOR	COSTMANOPE	PRETRETETA
N,6	N,5	N,5

b) Alege prin dialog și afișează pe ecran o rețetă solicitată înlocuind codurile produselor cu numele lor.

Soluție:

```
set talk off
clear
close all
use alimente
use retete in 2
defi menu m
defi pad p1 of m prompt 'Elaborare reteta'
defi pad p2 of m prompt 'Cautare-afisare reteta'
defi pad p3 of m prompt 'Iesire'
defi popup pop from 1,4 title 'Alegeti alimentul' prompt field
dena
defi popup afir from 1,14 title 'Alegeti reteta' prompt field
retete.numer
on sele pad p1 of m do creare
on sele pad p2 of m acti popu afir
on sele popu afir do afis
on sele pad p3 of m deac menu
acti menu m
close all
return
**********
proc creare
nume=spac(20)
@5,20 say 'Dati numele retetei:' get nume
read
sele retete
appe blan
```

Scanned with OKEN Scanner

```
sele alimente
stor 0 to tc,tp
i=1
on sele popup pop do adaug with tc,tp,i
acti popup pop
retu

proc afis
hide popu afir
i=1
cc=allt(str(i))
sele retete
plin=!empty(coda&cc)
@ 5,8 say 'Ingrediente:'
do while plin and (i<=5)
sele alimente
loca for coda=retete.coda&cc
@ 5+i,10 say dena+str(retete.canta&cc)
i=i+1
cc=allt(str(i))
sele retete
plin=iif(i<=5,!empty(coda&cc),.t.)
endd
@ i+7,10 say ' Calorii:'+str(totcalor)
@ i+8,10 say 'Pret ret:'+str(pretret)
@ i+9,10 say 'Manopera:'+str(costman)
wait wind 'Ai vazut'
clea
retu

proc adaug
para tc,tp,i
hide popu pop
sele retete
repl numer with nume
  cc=allt(str(i))
  @6,20 say 'Dati cantitatea pt. produsul: ';
                              +rtri(alimente.dena) get canta&cc
  read
  repl coda&cc with alimente.coda
  tc=tc+alimente.calorii*canta&cc
  tp=tp+alimente.pret_unit*canta&cc
  i=i+1
  @7,20 say 'Mai avem produse <d/n>?:' get mai defa 'D '
  read
  if uppe(mai)='D' and i<=5
    sele alimente
    show popu pop
    clea
```

Scanned with OKEN Scanner

```
  else
    sele retete
    clea
    @ 6,20 say 'Costul manoperei:' get costman
    read
    repl totcalor with tc, pretret with tp
    clea
    deact popu pop
  endi
return
```

12. Elevi

Se consideră două fişiere de date **ELEVI.DBF** şi **ADRESE.DBF** având structura:

ELEVI.DBF:

CLASA	NUME	MEDIE
C,3	C,20	N,5.2

ADRESE.DBF:

NUME	ADRESA
C,20	C,40

Să se scrie un program care realizează următoarele cerinţe:

a) Alege prin dialog numele anumitor elevi, în ambele fişiere. Pentru fiecare elev care se regăseşte se va afişa clasa, numele, adresa şi media.

b) Creează un fişier cu structura:

NUME	CLASA	MEDIE	ADRESA
C,20	C,3	N,5.2	C,40

pentru toţi elevii din primul fişier care se regăsesc în al doilea.

c) Afişează unul din cele trei fişiere (pe baza unei alegeri).

Soluţie:

```
set talk off
set head off
set safe off
close data
clea
use elevi
use adrese in b orde nume
set rela to nume into adrese
copy stru to eladr
sele c
use eladr
copy stru exte to temp    && crearea fisierului pt. pct. b
use temp
appe blan
```

Scanned with OKEN Scanner

```
repl field_name with 'adresa',field_type with 'c'
repl field_len with 40
crea eladr from temp
erase temp.dbf
appe from elevi          && incarcarea fisierului nou cu date
set rela to nume into b
repl all adresa with b.adresa
dele for empty(adresa)
      && stergerea articolelor care nu se regesesc in adrese.dbf
pack
sele a
defi menu me
defi pad p1 of me prom 'Selectarea fisierului'
defi pad p2 of me prom 'Listare'
defi pad p3 of me prom 'Iesire'
defi popu pp1 from 1,22 prom file like *.dbf
defi popu pp2 from 1,42
defi bar 1 of pp2 prom 'In mediu'
defi bar 2 of pp2 prom 'In DOS'
on sele pad p1  of me acti popu pp1
on sele popu pp1 do aleg
on sele pad p2 of me acti popu pp1
on sele pad p3 of me acti popu pp2
on sele bar 1 of pp2 do ies
on sele bar 2 of pp2 quit
acti menu me
retu
**********
proc ies
  close data
  set talk on
  clea
  deact menu all
retu
**********
proc aleg
  if pad()='P1'
    if at('ELEVI',prom())#0
      defi popu num from 7,25 prom fiel nume mess 'Alege elevul'
      on sele popu num do pcta
      acti popu num
    else
      defi popu num from 7,25 prom fiel b.nume;
                                          mess 'Alege elevul'
      on sele popu num do pctb
      acti popu num
    endi
```

Scanned with OKEN Scanner

```
  else
    do case
      case at('ELEVI',prom())#0
        sele a
      case at('ADRESE',prom())#0
        sele b
      case at('ELADR',prom())#0
        sele c
    endc
    defi wind w1 from 8,10 to 20,60
    acti wind w1
    disp all
    wait wind 'Apasa o tasta'
    sele a
    deact wind w1
  endi
retu

proc pcta
  hide popu num
  defi wind w1 from 8,10 to 20,65
  acti wind w1
  disp nume,clasa,medie,iif(empty(b.adresa),'nu are adr.',b.adresa) off
  wait wind 'Apasa o tasta'
  deact wind w1
  deact popu num
retu

proc pctb
  hide popu num
  defi wind w1 from 8,10 to 20,65
  acti wind w1
  cau=b.nume
  loca for nume =cau
  disp nume,clasa,medie,b.adresa off
  wait wind 'Apasa o tasta'
  deact wind w1
  deact popu num
retu
```

13. Mişcări

Se dă un fişier de date **MISCARI.DBF** cu structura:

CODP	DENP	UM	TIPM	CANTP	PRETU	DATAM
N,6	C,20	C,3	C,1	N,5	N,5	D

care conţine mişcările de produse finite ale unei societăţi. Câmpul `TIPM` indică operaţia de intrare/ieşire (I/O) a produsului, `DATAM` este data calendaristică la care are loc mişcarea (pot exista coduri diferite pentru acelaşi produs datorită faptului că au preţuri diferite).

Scanned with OKEN Scanner

Să se scrie un program care realizează următoarele cerințe:

a) Creează două fișiere noi: **INTRARI.DBF** și **IESIRI.DBF** care conțin pentru fiecare produs, suma cantităților intrate, respectiv suma cantităților ieșite.

Cele două fișiere vor avea structurile identice și anume:

CODP, DENP, UM, CANTP, VALOARE, DATAUM,

unde DATAUM reprezintă data ultimei mișcări a produsului, iar VALOARE suma valorilor produselor.

b) Afișează lista produselor din care nu s-a vândut nimic în ultimele 30 zile.

Soluție:

```
set talk off
set head off
set safe off
set date to brit
close all
clear
use miscari
copy stru exte to tmp
use tmp
appe blan
repl field_name with 'VALOARE',field_type with 'N',field_len with 10
crea mis from tmp
appe from miscari
replace all valoare with cantp*pretu
inde on codp tag cd
use
define menu meniu
define pad opt1 of meniu prompt 'Creare fisiere'
define pad opt2 of meniu prompt 'Afisare       '
define pad opt3 of meniu prompt 'Iesire        '
on sele pad opt1 of meniu do p1
on sele pad opt2 of meniu do p2
on sele pad opt3 of meniu do p3
activate menu meniu
retu
**********
procedure p1
  clear
  use mis order tag cd
  total to intrari on codp fields cantp,valoare for tipm='i'
  total to iesiri on codp fields cantp,valoare for tipm='o'
  use intrari
  index on codp tag codp ascending
  ? 'total intrari:'
  list off
  wait
```

Scanned with OKEN Scanner

```
  use iesiri
  index on codp tag codp ascending
  ? 'total iesiri:'
  list off
  wait
  use
return
**********
procedure p2
  clear
  close data
  use mis
  sort on codp, datam/d to miso for tipm='o'
  sele b
  use miso
  inde on codp tag cp
  sele a
  use iesiri
  set rela to codp into b
  repl all datam with b.datam
  ? 'produse fara vanzare in ultima luna:'
  ? 'data curenta:',date()
  list denp  for datam<gomonth(date(),-1) off
  wait
  use
return
**********
procedure p3
 close all
 deactivate menu
return
```

14. Medicamente

Să se elaboreze o aplicație care realizează o gestionare a stocului de medicamente dintr-o farmacie. Baza de date **MEDIC.DBF** are structura:

DENUMIRE	COD	UM	PRET	CANTITATE	DATA_EXP
C,20	C,6	C,3	N,5	N,5	D

Aplicația trebuie să gestioneze stocul de medicamente astfel:

a) La sosirea unui nou lot de medicamente se actualizează stocul existent (dacă un medicament nou sosit există în farmacie dar cu alt preț sau cu același preț dar cu data expirării diferită, se va crea o nouă înregistrare).

b) În momentul vânzării unui medicament stocul se diminuează cu cantitatea vândută. iar în cazul vânzării întregului stoc, medicamentul respectiv este șters din baza de date.

c) Listarea medicamentelor ce urmează să expire în luna curentă.

Scanned with OKEN Scanner

Soluție:

```
clear
set talk off
set date to brit
close all
use medic
define menu princ
define pad intra of princ prompt 'Intrari ' ;
message ' Intrari de medicamente in farmacie. Informatii '
define pad iesir of princ prompt 'Iesiri ' ;
message ' Iesire de medicamente din farmacie si informatii '
define pad liste of princ prompt ' Liste ' mess ' Liste cu ;
                                          medicamente  expirate'
define pad ies  of princ prompt 'Iesire'
  define popup e1 from 1,9 shad
  define bar 1 of e1 prompt 'Vanzare in stoc limitat'
  define bar 2 of e1 prompt '  Vanzare stoc total    '
on selection pad intra of princ do introd
on selection pad iesir of princ acti popu e1
  on selection bar 1 of  e1 do limit
  on selection bar 2 of  e1 do total
on selection pad liste of princ do exp1
on selection pad ies  of princ deactivate menu
activate menu princ
clear
return
**********
procedure introd
  clear
  nume=space(20)
  store 0 to mas,p
  c=space(6)
  um1=space(3)
  man={00/00/00}
  @ 10,5 say ' Codul medicamentului ' get c
  @ 11,5 say ' Ce medicament a sosit in farmacie?' get nume
  @ 12,5 say ' In ce cantitate se afla acest medicament?' get mas
  @ 13,5 say ' La ce pret se afla o UM ' get p picture '99999'
  @ 14,5 say ' Ce fel de UM se foloseste ' get um1
  @ 15,5 say ' In ce data  expira?' get man
  read
  locate for den=nume
  if not found()
    append blank
    replace cod with c,den with nume,um with um1,pret;
                          with p,cant with mas,data_exp with man
  else
    locate for pret=p
```

Scanned with OKEN Scanner

```
    if not found()
      append blank
      replace cod with c,den with nume,um with um1,pret;
                          with p,cant with mas,data_exp with man
    else
      if man<>data_exp
        append blank
        replace cod with c,den with nume,um with um1,pret;
                                              with p,cant with mas
      else
        clear
        @ 12,24 say ' Mai aveti acest medicament in farmacie '
        clear
        @ 12,6 say ' Acum vom actualiza stocul '
        replace cant with cant+mas
        clear
        @ 13,6 say ' Stocul a fost actualizat la: '+str(cant)+' '+um
      endif
    endif
  endif
  retu
**********
procedure limit
  clear
  denm=space(20)
  @12,5 say ' Din ce medicament doriti sa cumparati ' get denm
  read
  locate for den=denm
  if not found()
    @ 13,5 say ' Nu mai avem acest medicament.Reveniti '
    return
  else
    @ 13,5 say ' Din acest medicament mai avem : '+str(cant)
    cant1=0
    @ 14,5 say ' Ce cantitate doriti sa cumparati? ' get cant1
    read
    @ 15,5 say ' In total va costa:  ' +str(cant1*pret)
    @ 16,5 say ' Va multumim. Mai reveniti '
    wait
    clear
    @13,5 say 'Acum vom reactualiza medicamentul ce s-a vandut'
    replace cant with cant-cant1
    @ 14,5 say ' In stoc mai avem : '+str(cant)
    wait
    clear
  endif
  retu
**********
```

```
procedure total
  clear
  denmed=space(20)
  @ 13,5 say ' Cu ce va putem servi?  ' get denmed
  read
  locate for den=denmed
  if not found()
    @15,5 say 'Imi pare rau. Stocul este epuizat. Mai reveniti'
    return
  else
    @15,5 say 'Aveti de platit suma de : '+str(cant*pret)
    delete
    pack
    @ 16,5 say ' Stocul din acest medicament s-a terminat si'+;
                                ' a fost sters din baza de date '
    wait
    clear
return
**********
procedure exp1
  clear
  @ 12,5 say ' Ziua curenta este : ' +dtoc(date())
  @ 13,5 say ' Lista medicamentelor expirate este '
  disp for date() > data_exp
  wait
  clear
return
```

15. Furnizori

Se dă un fişier de date **MATERIAL.DBF** şi un fişier **FURNIZOR.DBF** având structurile:

MATERIAL.DBF:

CODM	DENM
N,6	C,15

FURNIZOR.DBF:

CODF	DENF	ADRESAF	CODM	PRETM
N,6	C,15	C,20	N,6	N,7.2

Să se scrie un program care realizează următoarele cerinţe:

a) Afişează toţi furnizorii unui material citit de la tastatură.

b) Determină pentru fiecare material, furnizorul cel mai convenabil (cu preţul materialului cel mai mic).

Soluţie:

```
set talk off
set head off
close data
clea
```

Scanned with OKEN Scanner

```
use furnizor orde pr
use material in b orde codm
set rela to codm into b
defi menu m1
  defi pad ale of m1 prom '\<Alege material'
  defi pad ie of m1 prom '\<Iesire'
  defi popu pp1  prom field b.denm
  on pad ale of m1 acti popu pp1
  on sele pad ie of m1 deacti menu m1
  on sele popu pp1 do afis
  acti menu m1
  close all
  retu
**********
proc afis
nums=promp()
sele b
loca for denm=nums
cods=codm
sele a
set filt to codm=cods
go top
defi wind w1 from 5,4 to 15,50 titl 'Furnizori'
max=pretm
if !eof()
  acti wind w1
  disp all denf,adresaf,pretm off
  wait wind 'Apasa o tasta pt. furniz. cel mai convenabil'
  @ 5,4 say 'Furnizorii cei mai convenabili'
  disp all denf,adresaf,pretm off for pretm=max
  wait wind 'Apasa o tasta pt. cont'
  deacti wind w1
else
  @ 7,20 say 'NU EXISTA FURNIZORI PT. PRODUSUL ALES'
  wait wind 'Apasa o tasta pt. cont'
endi
clea
retu
```

16. Produse

Se dă un fişier de date **MATERIAL.DBF** cu structura:

CODM	DENM	CANT	PRETU
N,6	C,15	N,4	N,5

Să se scrie un program care realizează următoarele cerinţe:

a) Adaugă un articol în fişierul **PRODUSE.DBF** pe baza alegerii prin dialog a materialelor care vor intra în componenţa fiecărui produs.

Structura fişierului **PRODUSE.DBF** va fi următoarea:

```
NUMEP  CODM1  CANTM1  CODM2  CANTM2  ...  CODM5  CANTM5  PRETTOT
 C,15   N,6     N,4     N,6     N,4        N,6     N,4     N,5
```

b) Afişează pe ecran, la alegere, un produs astfel constituit, codurile produselor fiind înlocuite cu numele lor.

Soluţie:

```
set talk off
clear
close all
use material in 1
use produse in 2
defi menu m
defi pad p1 of m prompt 'Elaborare produs'
defi pad p2 of m prompt 'Cautare-afisare produs'
defi pad p3 of m prompt 'Iesire'
defi popup pop1 from 1,1 title 'Alegeti materialul';
 prompt field denm foot 'Esc-Iesire'
defi popup pop2 from 1,20 title 'Alegeti un produs';
 prompt field numep foot 'Esc-Iesire'
on sele pad p1 of m do creare
on sele pad p2 of m acti popup pop2
on sele popup pop2 do afis
on sele pad p3 of m deac menu
acti menu m
clear
return
**********
procedure creare
  clear
  nume=spac(20)
  @4,29,6,71 box
  @5,30 say 'Numele produsului:' get nume
  read
  sele 2
  appe blank
  repl numep with nume
  i=1
  totp=0
  on sele popup pop1 do adaug
  acti popup pop1
  repl prettot with totp
  sele 1
retu
**********
```

Scanned with OKEN Scanner

```
procedure adaug
  if i<=5
    cant=0
    @8,25 say 'Material:'+prompt()
    @9,25 say 'Cantitate:' get cant pict '9999'
    read
    camp1=field(i*2)
    camp2=field(2*i+1)
    repl &camp1 with a.codm,&camp2 with cant
    totp=totp+cant*a.pretu
    i=i+1
  else
    wait wind 'Depasire nr.materiale!!!'
  endif
  clear
return
**********
procedure afis
  clear
  sele 2
  @8,50,10,79 box
  @9,51 say 'Produs : '+prompt()
  @12,49,22,78 box
  for i=1 to 5
    camp1=field(2*i)
    if &camp1=0
      exit
    endif
  camp2=field(2*i+1)
  sele 1
  locate for codm=b.&camp1
  sele 2
  @12+i,50 say a.denm+str(&camp2,10)
  next
  @21,50 say 'Cost produs  :'+str(prettot,10)
return
```

17. Cheltuieli de întreţinere

Se consideră un bloc cu două scări A şi B.

Să se calculeze cheltuielile de întreţinere ale locatarilor pentru fiecare apartament în parte. Cheltuielile de întreţinere cuprind:

– o factură pentru plata căldurii întregului bloc; plata se face în funcţie de suprafaţe;

– o factură pentru apa caldă consumată; plata se face în funcţie de numărul de persoane din fiecare apartament;

– o factură pentru apa rece consumată; se plăteşte în funcţie de numărul de persoane din fiecare apartament.

Scanned with OKEN Scanner

Soluție:

Programul utilizează baza de date **LOCATARI.DBF** cu următoarea structură:

NR_AP	SCARA	PERSOANE	CAMERE	CHELT
N,3	C,1	N,2	N,1	N,8

Programul rezolvă strict subiectul, în consecință am făcut următoarele presupuneri:
– baza de date este creată şi indexată în afara programului;
– pentru indexare a fost lansată comanda:

```
index on scara+str(nr_ap,3) tag sc_ap
```

– înregistrările au fost introduse în afara programului, prin program trebuie completat numai câmpul CHELT.

```
set console on
set talk off
set exact off
clear
close all
use locatari order sc_ap
@ 2,4 say 'Introduceti valorile facturilor'
@ 3,15 say 'caldura:    ' get cald default 0 valid cald>0 error;
                                                 'Factura incorecta'
@ 4,15 say 'apa calda: ' get apac default 0 valid cald>0 error;
                                                 'Factura incorecta'
@ 5,15 say 'apa rece: ' get apar default 0 valid cald>0 error;
                                                 'Factura incorecta'
read
sum persoane,camere to total_p,total_c
repl all chelt with;
              camere*cald/total_c+persoane*(apac+apar)/total_p
clear
do afisare with 'A'
do afisare with 'B'
**********
procedure afisare
  para sc
  clear
  defi wind indic at 20,5 size 1,45
  acti wind indic
  ? '            pentru iesire tastati <Esc>    '
  defi wind afis at 3,10 size 15,35 double title 'Scara '+sc
  acti wind afis
  brow noedit for scara=sc fiel nr_ap,persoane,camere,chelt;
                                                      in afis
  deac wind afis
  deac wind indic
return
```

Scanned with OKEN Scanner

18. Clasament

Să se întocmească un program care ține la zi clasamentul celor șase grupe ale Campionatului Mondial de Fotbal.

Se utilizează fișierele **ECHIPA.DBF** și **REZULTAT.DBF** cu structura:

ECHIPA.DBF:

NUMECH	GRUPA
C,30	C,1

REZULTAT.DBF:

ECHIPA_1	ECHIPA_2	GOLM1	GOLM2
C,30	C,30	N,2	N,2

unde: GOLM1 – numărul de goluri înscrise de echipa 1
GOLM2 – numărul de goluri înscrise de echipa 2

a) Să se creeze o bază de date care conține: numele echipei, grupa, numărul de meciuri jucate, numărul meciurilor câștigate, numărul meciurilor jucate la egalitate, numărul meciurilor pierdute, golaveraj (diferența dintre numărul de goluri date și numărul de goluri primite), punctaj (pentru victorie se acordă trei puncte, pentru meci egal un punct și nici un punct pentru meci pierdut);

b) Să se afișeze clasamentul pentru o grupă.

Soluție:

```
set talk off
set safe off
close all
clea
decla gr(6)
for i=1 to 6
  gr(i)=chr(64+i)
endf
gg=' '
@1,39 SAY 'Alege grupa:' get gg from gr
read
@0,39 SAY 'Grupa selectata:' + gg
@1,0 fill to  24,79
use echipa
set filt to grupa=gg
use rezultat in 2
sele 2
set filt to grupa=gg
use clasam in 3
sele 3
zap
sele 1
decla g[2],n[2]
```

Scanned with OKEN Scanner

```
defi menu meniu
defi pad p1 of meniu prompt 'Meci nou'
defi pad p2 of meniu prompt 'Clasament'
defi pad p3 of meniu prompt 'Iesire'
on sele pad p1 of meniu do mecinou
on sele pad p2 of meniu do clasam
on sele pad p3 of meniu deac menu
defi popup p1 title 'Alegeti o echipa' prompt field numech
on sele popup p1 do admeci
acti menu meniu
return
**********
procedure mecinou
@2,0 fill to  24,79
  i=0
  @8,30 say 'Joaca echipele :    Goluri:  '
  acti popup p1
@2,0 fill to  24,79
return
**********
procedure admeci
  i=i+1
  n[i]=prompt()
  if i=2 and n(1)=n(2)
    wait wind' NU SE POATE!! Reintroduceti cele doua echipe';
                                                          nowait
    i=0
    @2,0 fill to  24,79
    retu
  else
    go=0
    @8+i,30 say +n[i]+':' get go pict '99'
    read
    g[i]=go
    if i=2
      sele 2
      appe blank
      repl echipa_1 with n[1],echipa_2 with n[2]
      repl golm1 with g[1],golm2 with g[2],grupa with gg
      deac popup p1
    endif
  endi
return
**********
procedure clasam
defi wind clas from 2,2 to 23,79 titl 'Clasament grupa:'+gg
foot 'Pentru iesire <ESC>'
  sele 2
```

Scanned with OKEN Scanner

```
  scan
    do calcclas with golm1,golm2,echipa_1,echipa_2
    do calcclas with golm2,golm1,echipa_2,echipa_1
  endscan
  sele 3
  set order to punctgol descending
 if recc()#0
  acti wind clas
  brow in wind clas noedit
  zap
  deacti wind clas
 else
  wait wind 'Nu s-au jucat meciuri in grupa:'+gg nowai
endi
return
**********
procedure calcclas
para g1,g2,n1,n2
  sele 3
  set order to echipa
  seek n1
  if not found()
    appe blank
    repl echipa with n1
  endif
  repl mj with mj+1,gm with gm+g1,gp with gp+g2
  repl golaveraj with gm-gp
  if g1>g2
    repl mc with mc+1,punctaj with punctaj+3
  else
    if g1=g2
      repl me with me+1,punctaj with punctaj+1
    else
      repl mp with mp+1
    endif
  endif
  sele 2
return
```

19. Mijloace fixe

Să se scrie un program care realizează următoarele cerințe:

a) Creează o bază de date **MFIXE.DBF** care va conține informații despre mijloacele fixe dintr-o societate comercială:

DENUMIRE	NR_INV	AN_FAB	DFUNCT	VALINI	VALR	CASAT
C,20	N,9	N,4	N,2	N,10	N,10	N,3

unde:
– DFUNCT reprezintă durata de funcționare (în ani) a mijlocului fix cu denumirea DENUMIRE fabricat în anul AN_FAB, având numărul de inventar NR_INV;
– VALINI reprezintă valoarea inițială a mijlocului fix;
– VALR reprezintă valoarea rămasă a mijlocului fix;
– CASAT reprezintă numărul de luni peste DFUNCT.

Valoarea rămasă se calculează cu formula:

```
VALR = VALINI - (VALINI/DFUCT) * (AN_FAB - ANCURENT)
```

b) Realizează întreținerea fișierului (adăugare, ștergere și modificare);
c) Listează o situație cu mijloacele fixe care ar fi trebuit să fie casate.

Soluție:

```
use mfixe
clear
set talk off
set date to brit
define popup meniu from 10,10 title 'meniu'
define bar 1 of meniu prompt '\<adaugare'
define bar 2 of meniu prompt '\<stergere'
define bar 3 of meniu prompt '\<modificare'
define bar 4 of meniu prompt '\<calcul val ramasa'
define bar 5 of meniu prompt '\<listare'
define bar 6 of meniu prompt '\<listare mijloace fixe casate'
define bar 7 of meniu prompt '\<terminare'
on selection bar 1 of meniu do adaug
on selection bar 2 of meniu do sterg
on selection bar 3 of meniu do modif
on selection bar 4 of meniu do ramas
on selection bar 5 of meniu do list
on selection bar 6 of meniu do list1
on selection bar 7 of meniu do term
activate popup meniu
retu
**********
procedure term
  close all
  deactivate popup meniu
  clear popups
retu
**********
procedure adaug
  hide popup meniu
  append blank
  @2,2 say "introduceti:"
  @3,5 say "denumirea:" get wden default space(20)
  @4,5 say "nr_inventar:" get wnrinv default 0 pict "9999999999"
```

Scanned with OKEN Scanner

```
  @5,5 say "an_fabricatie:" get wanfab default 0 pict "9999"
  @6,5 say "durata_functionare(in ani): "get wdurata default 0;
                                                    pict "99"
  @7,5 say "valoare_initiala:" get wvalini default 0;
                                                  pict "9999999999"
  read
  replace denumire with wden
  replace nr_inv with wnrinv
  replace an_fab with wanfab
  replace dfunc with wdurata
  replace valini with wvalini
  wait
  clear
return
**********
procedure modif
  hide popup meniu
  clear
   @2,2 say "introduceti denumirea articolului care doriti sa-l;
                                                       modificati"
  @3,5 get wden default space(20)
  read
  go top
  locate for denumire=wden
  if .not. found()
    clear
    @ 3,2 say "articolul nu exista "
    wait
  else
    edit
  endif
  clear
retu
**********
procedure sterg
  hide popup meniu
  clear
  @2,2 say "introduceti denumirea articolului care doriti;
                                                  sa-l stergeti"
  @3,5 get wden default space(20)
  read
  go top
  locate for denumire=wden
  if .not. found()
    clear
    @ 3,2 say "articolul nu exista "
    wait
```

Scanned with OKEN Scanner

```
  else
    delete
    pack
  endif
  clear
retu

procedure ramas
  hide popup meniu
  clear
  @2,2 say "Denumirea articolului pt calcularea valorii ramase"
  @3,5 get wden default space(20)
  read
  go top
  locate for denumire=wden
  if .not. found()
    clear
    wait wind " articolul nu exista "
  else
    wvalram=0
    wvalram=valini - (valini/dfunc)*(year(date())-an_fab)
  endif
  replace valr with wvalram
  clear
retu
**********
procedure list
  hide popup meniu
  list
  wait
  clear
retu
**********
procedure list1
  hide popup meniu
  go top
  wan=0
  wan1=0
  do while .not. eof()
    wan=year(date())-(an_fab+dfunc)
    if wan=0
      replace casat with month(date())
    endif
    if wan>0
      wan1=month(date())+(wan)*12
      replace casat with wan1
    endif
    skip
  enddo
```

Scanned with OKEN Scanner

```
  set filter to casat > 0
  list
  wait
  set filter to
  clear
retu
```

20. Biblioteca

Să se scrie un program care realizează următoarele cerințe:

a) Creează un fișier **CARTI.DBF** care cuprinde informații despre cărțile dintr-o bibliotecă:

AUTOR	TITLU	DOMENIU	ANAP	EDITURA
C,20	C;20	C,20	N,4	C,15

b) Caută o carte după titlul aproximativ.

c) Afișează toate cărțile unei edituri care au apărut într-un anumit an în ordinea alfabetică a autorilor și în cadrul unui autor în ordinea alfabetică a titlurilor.

Soluție:

```
set talk off
set safe off
close all
clear
defi menu m
defi pad p1 of m prompt '\<Crearea bazei'
defi pad p2 of m prompt 'Cau\<tare carte'
defi pad p3 of m prompt '\<Afisare'
defi pad p4 of m prompt '\<Iesire'
defi wind f from 5,15 to 13,55 double title 'Carte
solicitata...'
on sele pad p1 of m do creare
on sele pad p2 of m do cautare
on sele pad p3 of m do afisare
on sele pad p4 of m deac menu
acti menu m
return
**********
procedure creare
  if not file('carti.dbf')
    create dbf carti (autor c(20),titlu c(20),domeniu;
                                  c(20),an_ap n(4),editura c(20))
    append
  else
    wait wind 'Baza deja exista!!!'
  endif
return
**********
```

Scanned with OKEN Scanner

```
procedure cautare
  use carti
  clear
  tit=spac(20)
  @5,10 say 'Dati titlul:' get tit
  read
  t=trim(uppe(tit))
  locate for uppe(titlu)=t
  if not found()
    wait wind 'Nu exista cartea '+t
  else
    do while found()
      clear
      acti wind f
      @1,1 say 'Titlu    :'+titlu
      @2,1 say 'Autor    :'+autor
      @3,1 say 'Domeniu  :'+domeniu
      @4,1 say 'Editura  :'+editura
      @5,1 say 'An apar. :'+str(anap,4)
      continue
      wait wind 'Next...'
      deac wind f
    enddo
  endif
return
**********
procedure afisare
  use carti
  index on editura+str(anap,4)+autor+titlu tag ind
  clear
  ed=spac(15)
  @5,10 say 'Dati editura  :' get ed
  an=0
  @6,10 say 'Anul aparitiei:' get an pict '9999'
  read
  locate for uppe(editura)=uppe(rtrim(ed)) and anap=an
    if not found()
    wait wind 'Nu exista in baza!!!'
  else
    defi wind f2 from 10,5 to 20,79 title 'Cartile editurii '+;
                        rtrim(ed)+'...'  foot 'Pt. iesire <ESC>'
    acti wind f2
    brow in wind f2 for uppe(editura)=uppe(rtrim(ed)) and;
                                                anap=an  noedit
    deacti wind f2
  endif
  clear
return
```

21. Actualizare simultană a două fişiere cu cerinţe speciale

Fie fişierele:

BEN.DBF

COD	NUME	COD_LOC
N,3	C,15	N,4

unde: COD – codul beneficiarului
NUME – numele beneficiarului
COD_LOC – codul localităţii beneficiarului

LOCA.DBF

COD_LOCA	NUME	TIP
N,4	C,15	C,1

unde: COD_LOCA – codul localităţii
NUME – numele localităţii
TIP – tipul localităţii (urban, rural)

Să se actualizeze fişierul BEN astfel încât să se asigure:
– pentru orice adăugare de articol, integritatea de identificare (câmpul COD unic);
– actualizarea simultană a fişierelor pornind de la fişierul BEN;
– integritatea de identificare pentru fişierul LOCA (pentru câmpul COD_LOC);
– garantarea integrităţii de referire între cele două fişiere.

Soluţie:

```
set talk off
close all
clea
use loca in b orde cod_loc
use ben
set rela to cod_loc into b
defi menu m1
defi pad p1 of m1 prom 'Intretinere';
                        mess 'Permite intretinerea fisierului'
defi popu pp1 from 1,7
defi bar 1 of pp1 prom 'Adaugare';
                       mess 'Permite adaugarea de noi articole'
defi bar 2 of pp1 prom 'Modificare';
                        mess 'Permite modificarea de articole'
defi bar 3 of pp1 prom 'Stergere';
                          mess 'Permite stergerea de articole'
defi pad p2 of m1 prom 'Iesire' mess 'Permite terminarea programului'
defi popu pp2 from 1,16
defi bar 1 of pp2 prom 'In dos'
defi bar 2 of pp2 prom 'In fox'
on sele pad p1 of m1 acti popu pp1
on sele bar 1 of pp2 quit
```

Scanned with OKEN Scanner

```
on sele bar 2 of pp2 do ies2
on sele pad p2 of m1 acti popu pp2
on sele bar 1 of pp1 do ada
on sele bar 2 of pp1 do mod
on sele bar 3 of pp1 do ste
acti menu m1
**********
proc ies2
  set talk on
  close data
  deacti menu m1
**********
proc ada
  defi wind w1 from 3,3 to 20,70
  acti wind w1
  d='D'
  do while uppe(d)='D'
    be=0
    nub=space(12)
    co=0
    @1,1 say 'Exista'+str(recc())+'articole'
    @3,5 say 'Codul beneficiarului:' get be pict '999' valid veri()
    @4,5 say 'Numele beneficiarului:' get nub
    @5,5 say 'Codul localitatii:' get co pict '9999'
    read
    append blan
    repl cod with be, nume with prop(nub), cod_loc with co
    if eof('b')
      nul=space(15)
      ti=' '
      @7,5 say 'Localitatea nu exista in fisier!!'
      @8,5 say 'Introduceti numele localitatii:' get nul
      @9,5 say 'Tipul localitatii (u-urban, r-rural)';
                                  get ti pict'!a' valid verit()
      read
      sele b
      appe blan
      repl cod_loc with co, nume with prop(nul);
                            tip with iif(ti='R','rural','urban')
      clea
      sele a
    endi
    @11,10 say 'Mai adaugati (d/n):' get d
    read
  endd
  deacti wind w1
retu
**********
```

```
function veri
  loca for cod=be
return iif(found(),.f.,.t.)
**********
function verit
  if ti='U' .or. ti='R'
    return .y.
  else
    return .n.
**********
procedure mod
  sele a
  defi popu mod from 3,19 prom fiel str(cod)+nume;
                          mess'Alegeti articolul de modificat;
                           si <Return> sau <Esc> pentu iesire'

  on sele popu mod do modp
  acti popu mod
return
**********
procedure modp
  rec=recn()
  defi wind w1 from 3,3 to 20,70
  acti wind w1
  be=cod
  nub=nume
  co=cod_loc
  @3,5 say 'Codul beneficiarului:' get be pict '999' valid verim()
  @4,5 say 'Numele beneficiarului:' get nub
  @5,5 say 'Codul localitatii:' get co pict '9999'
  read
  repl cod with be, nume with prop(nub), cod_loc with co
  if eof('b')
    nul=space(15)
    ti=' '
    @7,5 say 'Localitatea nu exista in fisier!!'
    @8,5 say 'Introduceti numele localitatii:' get nul
    @9,5 say 'Tipul localitatii (u-urban, r-rural)' get ti;
                                          pict '!a'valid verim()

    read
    sele b
    appe blan
    repl cod_loc with co, nume with prop(nul);
                          tip with iif(ti='R','rural','urban')
    clea
    sele a
  endi
  deacti wind w1
return
```

Scanned with OKEN Scanner

```
function verim
  loca for cod=be
  if not found()
    nas=.t.
    go rec
  else
    go rec
    dele
    set dele on
    loca for cod=be
    nas=not found()
    reca all
    go rec
  endi
return nas

procedure ste
  sele a
  defi popu ste from 3,19 prom fiel str(cod)+nume;
        mess 'Alegeti articolul de sters si <Return> sau <Esc>;
                                                pentru iesire'
  on sele popu ste do step
  acti popu ste
return

procedure step
  rec=recn()
  vv='D'
  sl='D'
  defi wind w1 from 3,3 to 20,70
  acti wind w1
  @3,5 say 'Codul beneficiarului:'+str(cod)
  @4,5 say 'Numele beneficiarului: '+nume
  @5,5 say 'Codul localitatii:'+str(cod_loc)
  @1,30 say 'Acesta este articolul d/n?:' get vv
  read
  if uppe(vv)='D'
    sa=recn()
    sab=recn('b')
    cl= cod_loc
    loca for cod_loc=cl and recn()#sa
    if not found()
      @7,5 say 'Doriti sa stergeti si localitatea d/n?:'get sl
      read
      if uppe(sl)='D'
        sele b
        go sab
        dele
        sele a
```

Scanned with OKEN Scanner

```
      endi
    endi
  endi
  go sa
  dele
  clea
  sele b
  pack
  sele a
  pack
  deacti wind w1
return
```

22. *Evidența candidaților*

Să se scrie un program pentru evidența candidaților la un concurs cu n probe (n se citește).

Programul trebuie să permită notarea elevilor, calcularea mediei, afișarea listei reușiților în ordinea descrescătoare a mediilor și alfabetic, lista respinșilor în ordine alfabetică. Numărul de locuri se citește de la tastatură.

Se dă fișierul **ELEV.DBF** cu următoarea structură:

NUME C,25 – conține numele elevului, inițiala tatălui și prenumele.

Acestui fișier i se va completa structura (automat) cu următoarele câmpuri:

NOTA1, NOTA2,..., NOTAn (n citit), MEDIA.

Soluție:

– pentru completarea structurii s-a folosit comanda COPY STRU EXTE și funcțiile standard STR(), LTRIM();

– completarea notelor s-a făcut folosind macrosubstituția și funcția standard FIELD()

– pentru afișarea candidaților admiși în ordinea descrescătoare a mediilor s-a folosit comanda SORT iar pentru cei în ordine alfabetică s-a folosit INDEX.

```
set talk off
set safe off
clear
close all
use elev
inpu 'Nr de probe:' to p && crearea fisierului cu n campuri NOTA
copy stru exte to xx
sele b
use xx
for i=1 to p
  appe blank
  replace field_name with 'nota'+ltrim(str(i,2)),field_type;
  with 'n',field_len with 5,field_dec with 2
endf
```

Scanned with OKEN Scanner

```
appe blank
replace field_name with 'media',field_type with 'n'
replace field_len with 5,field_dec with 2
create elevi from xx
use elevi
appe from elev
go top
scan   && notarea elevilor si calcularea mediei
  clea
  @3,3 say 'Notele pt: '+allt(nume)+' sunt'
  s=0
  for i=1 to p
    n=0
    @4,3 say 'Nota:'+ltrim(str(i,2)) get n  picture '99.99'
    read
    s=s+n
    x=field(i+1)
    replace &x with n
  endf
  replace media with s/p
endscan
sort on media/d,nume to adm
use adm
* afisarea candidatilor admisi in ordinea descrescatoare a mediilor
inpu 'Nr de locuri:' to l
@3,3 say 'ADMISII IN ORDINEA DESCRESCATOARE A MEDIILOR:'
disp all while recno()<=l off
wait wind "ADMISII DUPA MEDII"
skip-1
k=media
skip
clea
if media=k && eventualele locuri suplimentate
  @ 3,3 say "SUPLIMENTARI"
  disp rest while media=k off
  wait wind "SUPLIMENTATII"
endi
r=recno()-1 && afisarea  candidatilor admisi in ordine alfabetica
set filt to recno()<=R
@3,3 say "ADMISII IN ORDINE ALFABETICA"
index on nume tag n
disp all off
count to x
wait wind "ADMISI ALFABETIC"
clea
set filt to
go x && afisarea candidatilor respinsi in ordine alfabetica
kk=media
```

Scanned with OKEN Scanner

```
@3,3 say " RESPINSII"
disp for media<kk
wait wind "RESPINSII"
set head on
set talk on
set safe on
retu
```

23. *Planificarea consultațiilor la un cabinet particular*

Programul trebuie să permită înscrierea pacienților (evident modificări, anulări) și informări pentru pacienți și medic despre zilele și orele rezervate.

Programul va folosi fișierul **PACIENT.DBF** cu următoarea structură:

```
NUME    ZIUA    ORA
C,20    C,10    N,2
```

Se presupune că medicul are programul cuprins între orele 16–20.

Soluție:

– pentru o nouă programare (adăugare) se va folosi comanda `@ say get` valorile citindu-se direct în câmpurile bazei de date;

– pentru operațiile de modificare și ștergere trebuie să se cunoască numărul articolului care trebuie modificat sau șters;

– în cadrul operației de ștergere există și posibilitatea renunțării la ștergerea articolului selectat; se folosesc comenzile DELETE, PACK, RECALL ș.a.;

– pentru vizualizarea programărilor dintr-o anumită zi se va citi într-o variabilă numărul zilei, se caută articolele care îndeplinesc acea condiție și se afișează.

```
set talk off
set safe off
clear
close all
public cc,x
use pacient
defi wind w1 from 5,2 to 20,75 title "PLANIFICARE CONSULTATII";
                        footer "CABINET PARTICULAR" shadow double
acti wind w1
defi menu pa && definirea meniului
defi pad a of pa prompt "actualizare";
mess "adaugari,modificari,stergeri de pacienti"
defi pad c of pa prompt "informari" mess "cine si cand vine"
defi pad d of pa prompt "iesire"
defi popu p1
defi bar 1 of p1 prompt "adaugari"
defi bar 2 of p1 prompt "modificari"
defi bar 3 of p1 prompt "stergeri"
on sele bar 1 of p1 do adaug
```

Scanned with OKEN Scanner

```
on sele bar 2 of p1 do modific
on sele bar 3 of p1 do sterg
on sele pad a of pa acti popu p1
on sele pad c of pa do informari
on sele pad d of pa do ies
acti menu pa
retu
**********
procedure ies && iesirea din program;revenirea in fereastra Commmand
  deacti wind all
  deacti menu
retu
**********
procedure adaug && procedura de adaugare a pacientilor in fisier
  defi wind w2 from 10,25 to 20,75 title "ADAUGARI"
  acti wind w2
  appe blan
  @ 2,2 say "nume:" get nume func "a"
  @ 3,2 say "ziua:" get ziua range 1,31
  @ 4,2 say "ora: " get ora range 16,20
  read
  disp off
  wait wind "tu esti"
  clea
  deacti wind w2
retu
procedure modific && modifica continutul campurilor din fisier
  defi wind w3 from 10,25 to 20,75 title "MODIFICARI PACIENTI"
  acti wind w3
  v='D'
  do while upper(v)="D"
  disp all
  wait wind "gaseste-ti numarul articolului!"
  clea
  do mo
  acce"continuati?" to v
  endd
  deacti wind w3
retu
**********
procedure mo
  defi wind w2 from 10,25 to 20,75 title "MODIFICARI"
  acti wind w2
  @ 2,2 say "numarul articolului de modificat:" get n defa 0;
                                        pict "999" range 1,recc()
  read
  loca for recn()=n
  clea
```

Scanned with OKEN Scanner

```
  @ 2,2 say "nume:" get nume func "a"
  @ 3,2 say "ziua:" get ziua range 1,31
  @ 4,2 say "ora: " get ora range 16,20
  read
  disp off
  wait wind "e modificat"
  deacti wind w2
retu
**********
procedure sterg && procedura de anulare a unei programari
  defi wind w4 from 10,25 to 20,75 title "STERGERE PACIENTI"
  acti wind w4
  mai='D'
 do while uppe(mai)='D'
  clea
  disp all
  wait wind "gaseste-ti numarul articolului!"
  clea
    @ 2,2 say "numarul articolului de sters:" get n defa 0;
                                        pict "999" range 1,recc()
  read
  disp reco n
  acce "acesta e articolul?(d/n)" to x
  if upper(x)="D"
    dele
    acce "sigur vrei sa-l stergi(d/n)?" to y
    if upper(y)="D"
      pack
      clea
      disp all
      wait wind "s-a efectuat stergerea!"
    else
      recall
      clea
      disp all
      wait wind "am renuntat!"
    endi
    endi
    acce 'Mai stergeti<d/n>:' to mai
 endd
  deacti wind w4
retu
**********
procedure informari && procedura de informare asupra timpului ocupat
  Z=0
  @ 3,3 say "ziua pentru care doriti afisarea pacientilor:";
                                              get z PICT "99"
  read
```

Scanned with OKEN Scanner

```
  index on str(ziua)+str(ora) tag zo
  loca for ziua=z
  defi wind w5 from 10,25 to 20,75;
                  title " pacientii din ziua "+ALLT(str(z)) shad
  acti wind w5
  if found()
    set head off
    ?"      NUME                ORA"
    disp while ziua=z fiel nume,ora off
    wait wind "EI SUNT"
  else
    ?"NU SUNT PACIENTI PLANIFICATI PENTRU ACEASTA ZI"
    wait wind "nu sunt"
  endi
  clea
  deacti wind w5
retu
```

24. Farmacii

Se dau următoarele fişiere:

FARM.DBF

COD	DEN	ADRESA	TEL
N,2	C,15	C,20	N,6

MED.DBF

COD	DENM	CANT	PRETU
N,2	C,15	N,5	N,&

unde:

- COD – codul farmaciei,
- DEN – numele farmaciei,
- ADRESA – adresa farmaciei,
- TEL – telefonul farmaciei,
- DENM – denumirea medicamentului,
- PRETU – preţul unitar al medicamentului,
- CANT – cantitatea existentă din acel medicament.

Se citeşte un medicament (prin denumirea sa) şi cantitatea necesară din acel medicament. Să se afişeze 3 farmacii (prin denumire, adresă şi telefon) care au medicamentul respectiv în cantitatea dorită şi cu preţul cel mai mic.

Soluţie:

– se ordonează articolele din baza de date **MED.DBF** după denumire, preţ şi cantitate folosindu-se comanda INDEX după care se parcurge fişierul.

```
set talk off
set escape on
set safe off
clea
```

Scanned with OKEN Scanner

```
close all
sele 0
use farm
set order to cod
sele 0
use med order denm
defi wind w1 from 5,5 to 20,75 shad title "FARMACII";
footer "IESIREA CU <ESC>, CONTINUARE CU <ENTER>"
acti wind w1
on escape  do kk
defi popu p1 prompt field denm
on sele popup p1 do proc1
acti popu p1
deacti popu p1
set talk on
set safe on
retu

procedure proc1
c=0
m=prompt()
@ 2,25 say "cantitatea necesara:      " get c pict "99999"
read
index on denm+str(pretu)+str(cant) tag pc
set rela to cod into farm
go top
loca for allt(upper(denm))=allt(uppe(m)) and cant>=c
if found()
hide popu p1
  @ 5,5 say "FARMACIILE CARE AU MEDICAMENTUL RESPECTIV"+;
                              " LA PRET MINIM SUNT:" colo g+/b+
  @ 7,7 say "NUME FARMACIE       ADRESA        TELEFON     PRETU "
  i=1
  do while NOT EOF() and i<=3
    ?"     ",a.den,"    ",allt(a.adresa),"    ",a.tel," ",pretu
    i=i+1
    cont
  enddo
  wait wind "ACESTEA SUNT"
else
  @ 10,5 say "NU EXISTA MEDICAMENTUL "+allt(m)+;
                                      " IN CANTITATEA CERUTA"
  wait wind "NU EXISTA"
endi
set rela to
set order to denm
clea
show popu p1
retu
```

Scanned with OKEN Scanner

```
proc kk
  deacti wind w1
  deacti popu p1
retu
```

25. *Agenda de lucru*

Se dă fişierul **ZI.DBF**:

LUNI	MARTI	MIERCURI	JOI	VINERI	SAMBATA	DUMINICA
M	M	M	M	M	M	M

Fişierul este completat prin program cu 24 de articole corespunzătoare celor 24 de ore ale unei zile.

Programul permite introducereă (respectiv modificarea) orarului unei zile, alegându-se ziua din popup, iar ora citindu-se într-o variabilă. Vizualizarea programului unei zile se face selectând din popup numele zilei, apoi într-o fereastră se va afişa ora şi activitatea care este înscrisă.

Soluţie:

```
set talk off
close all
clea
publ b,w1,w2
use zi
if recc()=0
  do procinc
else
  defi wind w1 from 5,1 to 20,75 title "AGENDA "shadow double
  defi wind w2 from 4,45 to 20,75 footer "IESIRE CU CTRL+W"
  acti wind w1
  defi menu zi in wind w1
  defi pad a of zi promp '\<actualizare';
                         mess 'permite completarea fisierului'
  defi pad b of zi promp '\<vizualizare';
                  mess "permite consultarea progamului pe zile"
  defi pad c of zi promp '\<iesire'
  defi popup p1 promp stru from 2,3
  on sele pad a  of zi acti popu p1
  on sele popu p1 do adaug
  defi popu p5 prom stru from 2,17
  on sele pad b of zi do zile
  on sele pad c of zi do ies
  acti menu zi
endi
retu
*********
```

Scanned with OKEN Scanner

```
procedure adaug
  p=prom()
  @ 5,25 say "dati nr. orei" get ora defa 0 pict '99'
  read
  go ora
  acti wind w2
  modi memo &p wind w2
  deacti wind w2
  acti wind w1
  clea
  deacti popu p2
retu
procedure procinc && initializarea celor 24 de ore ale unei zile
  i=1
  do while i<=24
    appe blan
    i=i+1
  endd
  wait wind
  clea
  go top
procedure zile && Afisarea programului unei zile
  on sele popu p5 do pp
  acti popu p5
  deacti popu p5
retu
procedure pp
  b=bar()
  go b
  x=field(b)
  go top
  i=1
  defi wind xx from 5,30 to 20,75 panel
  acti wind xx
  do while not eof()
    if not empty(&x)
      y=&x
      @ i,1 say str(recno(),3)+" - "+str(recno()+1,2)+" "+y
      i=i+1
      if i>=13
        wait wind 'asteapta 3 sec' time 3
        clea
        i=1
      endi
    endi
    skip
  endd
```

Scanned with OKEN Scanner

```
  wait wind "Programul la ora aleasa"
  deacti wind xx
retu to master

procedure ies
  deacti wind all
  deacti menu
retu
```

26. *Combinare de fişiere*

Se dau două baze de date:

PRODUSE.DBF cu structura:

COD_PROD	CANTITATE	PRET_UNIT
N,9	N,10.3	N,11

NOMENC.DBF cu structura:

COD_PROD	DENPROD
N,9	C,20

Să se realizeze un fişier rezultat din combinaţia fişierelor date, astfel încât să se poată lista simultan toate câmpurile celor două fişiere.

Soluţie:

Se va folosi comanda `JOIN` care realizează fişierul având câmpurile comune celor două baze.

```
close all
set talk off
set safe off
sele 1
use produse
sele 2
use nomenc
sele 1
join with nomenc to prod for cod_prod=nomenc.cod_prod
use prod
lp=20
clear
@ 1,15 say 'Lista Produselor la Data:'+DTOC(DATE())
do capt10
lc=6
store 0 to v1,v2
nr_crt=1
scan
  @lc,1 say '|'+str(nr_crt,3)+'|'+str(cod_prod,9)+'|'+;
                          denprod+'|'+str(cantitate,10,3)+'|'+;
        str(pret_unit,11)+'|'+str(cantitate*pret_unit,18,2)+'|'
  lc=lc+1
  if lc>lp
    nr_crt=1
    wait
    do capt10
```

```
    lc=6
  endif
  nr_crt=nr_crt+1
  v1=v1+cantitate
  v2=v2+cantitate*pret_unit
endscan
@lc,1 say '_____________________________________________________'
@lc+1,1 say '|Valori Finale |'+str(v1,10,3)+'|';
+str(v2,18,2)+ '|'
@lc+2,1 say '_____________________________________________________'
***********
procedure capt10
  @3,1 say '_____________________________________________________'
  @4,1 SAY '|NR.|COD PROD.|        DENUMIRE        |CANTITATE |  ;
                                          PRET/U   |  VALOARE   |'

  @5,1 SAY '_____________________________________________________'
return
```

27. *La bancă*

Într-o bancă se foloseşte pentru înregistrări, fişierul:

BANCA.DBF având câmpurile

DATA	SUMADISP	DEPUNERI	COD	SOLD
D,8	N,10	N,10	C,4	N,10

Să se realizeze o aplicaţie de exploatare a fişierului care respectă condiţiile:

– nu se pot modifica decât clienţii înregistraţi în nomenclatorul **CLIENTI.DBF** care trebuie relaţionat cu **BANCA.DBF**;

– codul clientului ce trebuie modificat trebuie să existe în **BANCA.DBF**;

– citirea datelor existente pentru un client se realizează pe o machetă care va conţine câmpurile SUMADISP, DATA, DEPUNERI şi soldul calculat cu SUMADISP+DEPUNERI

Soluţie:

```
clear
set talk off
set safety off
sele 1
use banca
sele 2
use clienti
index on cod to i2
sele 1
set relation to cod into b
mai='D'
```

```
do while uppe(mai)='D'
  clear
  c=space(4)
  @1,1 say 'EXPLOATARE'
  @2,1 say 'TASTATI CODUL CLIENTULUI' get c valid va()
  read
  locate for cod=c
    repl data with date()
      @10,5 clear
      @11,5 say 'Clientul :' get clienti.client disable
      @12,5 say 'Suma disponibila:' get sumadisp disable
      @13,5 say 'Suma depusa:' get depuneri
      @14,5 say 'Data operatiunii:' get data disable
      read
      s=sumadisp+depuneri
      replace sold with s, depuneri with 0
      @14,5 say 'Sold          .    :' get sold disable
      wait ''
  @15,10 say 'Mai doriti pt. alt client:' get mai
  read
enddo

function va
  sele 2
  if not seek(c) and not empty(c)
    m=1
    defi popu p1 from 3,1 title 'Clienti Inregistrati';
    prompt fiel cod+''+client
    @4,1 say '' get m popup p1
    read
    c=cod
  endif
  sele 1
  @3,1 clear
return.t.
```

28. Personal (1)

Se dau bazele de date ale personalului unei firme cuprinse în fişierele **PERS.DBF** şi respectiv **LMUNCA.DBF**.

Cele două baze au structura după cum urmează:

PERS.DBF:

MARCA	NUME	LOCM	SALARIU	DATA_VECM	SPOR_V
N,6	C,26	N,5	N,7	N,6	N,2

LMUNCA.DBF:

LOCM	DEN_LOCM
N,5	C,20

Scanned with OKEN Scanner

Se cere să se listeze la imprimantă un tabel cuprinzând, în ordine alfabetică, următoarele câmpuri:
– marca
– numele şi prenumele
– locul de muncă
– salariul
– sporul de vechime
– vechimea
– total salariu plus spor vechime

Soluţie:

```
set talk off
dele file per.idx
set device to screen
clea
data=date()
@ 12,12 say 'Tastati data ' get data
read
set device to print
set print on
l=1
s1=0
s2=0
sele 1
use pers
index on subs(str(locm,5),1,2)+nume to per
do while .not. eof()
  a=locm
  l=l+1
  @ 1,15 say 'TABEL NOMINAL CUPRINZAND SALARIILE PERSONALULUI'
  l=l+1
  sele 2
  use lmunca index lmunca
  seek a
  @ 1,20 say den_locm
  l=l+2
  @ 1,1 say repl('=',97)
  l=l+1
  @ 1,1 say '|Nr.crt| Marca | Nume si prenume |Loc mun|;
                  Salariu | Data_spor | Vech% | Vech |  Total  |'
  l=l+1
  @ 1,1 say repl('=',97)
  sele 1
  sal=0
  i=1
```

```
  do while not eof() and locm=a
    if l>90
      l=1
    endif
    l=l+1
    @ l,1 say '|'+str(i,4)+'|'+str(marca,6)+'|'+nume+;
                        '|'+str(locm,5)+'|'+str(salariu,7)+'|'+;
    str(data_vecm,4)+'|'+str(spor_v,2)+'|';
                                  +str(spor_v*salariu/100,6)+'|'+;
    str(salariu*(1+spor_v/100),7)+'|'
    i=i+1
    sal=sal+salariu
    s1=s1+spor_v*salariu/100
    s2=s2+salariu*(1+spor_v/100)
    skip
  enddo
  l =l+1
  @ l,1 say repl('-',97)
  l=l+2
  @ l,1 say ' TOTAL'+spac(43)+str(sal,8)+spac(18)+;
                                          str(s1,8)+str(s2,10)
  store 0 to sal,s1,s2
  l=l+1
  @ l,1 say repl('=',97)
  l=l+4
  @ l,15 say ''+spac(17)+''
  l=l+1
  @ l,15 say ''
  l=1
  wait wind 'Alt loc de munca/sectie'
enddo
close all
l=l+1
@ l,1 say ' '
set print off
set device to screen
return
```

29. Personal (2)

Se dau bazele de date **PERSONAL.DBF** şi **RETPERM.DBF**. În fişierul **PERSONAL.DBF** se găsesc datele personale ale angajaţilor unei societăţi iar în fişierul **RETPERM.DBF** sunt reţinerile lor (chirii, imputaţii, rate locuinţe, CAR, garanţii materiale etc.) reprezentate prin codurile lor astfel:

- 201 – chirii IGO
- 202 – imputaţii
- 203 – rate locuinţe
- 204 – popriri
- 205 – rate OCL
- 206 – CAR

Scanned with OKEN Scanner

Să se realizeze o listă (raport) pe categorii de reţineri care să cuprindă următoarele câmpuri:
– Numele şi prenumele
– codul reţinerii
– rata

Cele două fişiere se vor crea după cerinţele problemei.

Soluţie:

```
set talk off
clea
close all
set safety off
use personal
index on marca tag marc
sele b
use retperm
set rela to marca into personal
dimension tr(7)
for i=1 to 7
tr(i)=200+i
endf
a=1
mai='D'
do while upper(mai)='D'
  @ 1,40 say 'Alegeti codul retinerii: '
  @ 1,66 get a from tr
  read
  a1=200+a
  l=1
  set filter to cod_ret=a1
  go top
  defi wind lis from 2,2 to 20,60
  acti wind lis
  do while not eof()
    @l,1 say str(marca,6)+'  '+a.nume+str(cod_ret,4)+;
                                                  str(rata,8)

    l=l+1
    skip
  endd
  acce 'Mai doriti?<d/n>:' to mai
  deacti wind lis
endd
clea
close all
return
```

30. Elevi

Se consideră fişierul **ELEVI.DBF** cu structura:

CLASA	NUME	NOTA1	NOTA2
C,6	C,15	N,5.2	N,5.2

Să se scrie un program care afişează:

NUME, NOTA1, NOTA2, MEDIA=(NOTA1+NOTA2)/2

pentru fiecare elev, calculează şi afişează media fiecărei clase pentru câmpurile NOTA1, NOTA2, MEDIA, calculează şi afişează media întregii şcoli pentru acele câmpuri.

Soluţie:

```
close all
set talk off
set safe off
use elevi
index on clasa+nume to i1
opt=' '
@1,1 say 'Listare pe E-ecran/I-imprimanta' get opt pict'!';
valid opt$ 'EI'
read
if opt='I'
  lp=66
  set device to print
  set print on
else
  clear
  lp=20
endif
pag=1
?'Situatia scolara pe clase la data de:'+DTOC(DATE())
lc=6
store 0 to ta,tb,tc,tc1
do while .not.eof()
store 0 to nrcrt,a,b,c1
  lc=lc+1
  c=clasa
  do capt14
  do while not eof() and clasa=c
    nrcrt=nrcrt+1
    ?'|'+str(nrcrt,3)+' |'+clasa+'   | ;
                          '+substr(nume,1,16)+'|'+str(nota1,5,2);
          +'|'+str(nota2,5,2)+'|'+str((nota1+nota2)/2,5,2)+'|'
    lc=lc+1
    a=a+nota1
    b=b+nota2
    c1=c1+(nota1+nota2)/2
```

Scanned with OKEN Scanner

```
    do paginare
    skip
  enddo
  ?'|Media clasei '+c+str(a/nrcrt,5,2)+'|'+;
                   str(b/nrcrt,5,2)+'|'+str(c1/nrcrt,5,2)+'|'

  ta=ta+a
  tb=tb+b
  tc1=tc1+c1
  lc=lc+1
  do paginare
enddo
?'|Media scolii'+space(6)+'  |'+str(ta/recc(),5,2)+'|'+'|'+;
                 str(tb/recc(),5,2)+'|'+str(tc1/recc(),5,2)+'|'
?'-------------------------------------------------'
? space(30)+'Pag:'+str(pag,3)
if opt='I'
  set print off
  set device to screen
endif

procedure paginare
  if lc>lp
    ?
    ?space(30)+'Pag:'+str(pag,3)
    wait ''
    clear
    do capt14
    lc=6
    pag=pag+1
  endif
return

procedure capt14
  ?'-------------------------------------------------'
  ?'|Nr. |Clasa|Nume si prenume  |Nota1 |Nota2 |Media |'
  ?'-------------------------------------------------'
return
```

31. Reţineri CAR

Să se realizeze o listă cu reţinerile la CAR ale personalului unei societăţi comerciale. Programul va folosi bazele de date **PERSONAL.DBF**, **RETPERM.DBF** şi **CAR.DBF**. În fişierul **PERSONAL.DBF** sunt reţinute datele personale ale angajaţilor, în fişierul **RETPERM** sunt păstrate reţinerile, iar în fişierul **CAR.DBF** este trecută rata la CAR.

Tabelul va avea următoarele câmpuri:

- MARCA (din fişierul **PERSONAL.DBF**)
- numele şi prenumele (din fişierul **PERSONAL.DBF**)

Scanned with OKEN Scanner

– RET (din fişierul **RETPERM.DBF**)
– RATA (din fişierul **CAR.DBF**)

Fişierul **PERSONAL.DBF** conţine:

MARCA	NUME	LOCM
N,6	C,26	N,5

Fişierul **RETPERM.DBF** conţine:

MARCA	COD_RET	RET
N,6	N,3	N,6

Fişierul **CAR.DBF** conţine:

MARCA	RATA
N,5	N,6

Observaţii:
– Codul reţinerii la CAR este 206.
– Lista se va realiza pe locuri de muncă (cod secţie).
– La sfârşitul sublistei corespunzătoare unui loc de muncă se vor totaliza reţinerile, respectiv ratele CAR.

```
set talk off
clear
use personal
index on str(locm)+nume tag  ln
sele b
use retperm
index on marca tag mar
sele c
use car
index on marca tag mar
sele a
set rela to marca into b
sele b
set rela to marca into c
sele a
l=1
set device to print
set print on
do while not eof()
  if b.cod_ret=206
  wrata=0
  wrat=0
  a=locm
  l=1
  @ 1,10 say 'RETINERI C.A.R.'
  l=l+1
  @ 1,1 say repl('-',60)
```

Scanned with OKEN Scanner

```
  l=l+1
  @ l,1 say  ' MARCA          NUME      SALARIU    RATA RETINUT'
  l=l+1
  @l,1 say repl('-',132)
  l=l+1
  do while a=locm
   if b.cod_ret=206
    if l>20
      l=0
    endif
      l=l+1
        @ l,1 say str(marca,6)+'    '+nume+;
                  str(salariu,7)+str(b.ret,7)+str(c.rata,7)
      l=l+1
      @ l,1 say repl('-',132)
      wrata=wrata+b.ret
      wrat=wrat+c.rata
   endi
   skip
  enddo
  l=l+3
  @ l,1 say 'TOTAL  '+spac(37)+str(wrata,7)+str(wrat,7)
  wait
  else
  skip
  endi
  clea
enddo
close all
l=l+1
@ l,1 say ' '
set print off
set device to screen
return
```

32. Test psihologic

Să se scrie un program pentru realizarea unui test psihologic. Fiecare întrebare are mai multe variante de răspuns. Pentru fiecare variantă se primeşte un anumit punctaj. Pentru o întrebare se poate alege doar un singur răspuns. Testul se interpretează de program în funcţie de punctajul obţinut şi se afişează.

Soluţie:

Pentru realizarea programului s-au conceput următoarele baze de date:

INTR.DBF

```
 CODI    INTREBARE
  N,2       C,200
```

VARIANTE.DBF

CODI	VARIANTE	PUNCTAJ
N,2	c,35	N,2

– dacă întrebarea cu codul x are n variante de răspuns, în **VARIANTE.DBF** vor exista n înregistrări pentru care CODI este x, fiecare variantă având un anumit punctaj.

INTERPR.DBF

LIMINF	LIMSUP	INTERPRET
N,2	N,2	M

– pentru punctajul realizat , se caută înregistrarea corespunzătoare în această bază de date şi se afişează conţinutul câmpului memo Interpret.

```
set safety off
set talk off
close all
clear
use intr
use variante in 2
use interpr in 3
total=0
define wind w1 from 1,1 to 23,70 title 'Test'
acti wind w1
sele 1
index on codi tag c
sele 2
index on codi tag c1
sele 1
set rela to
set relation to codi into 2
set skip to 2
go top
o=' '
do while not eof()
  crt=recno()
  clear
  rev=.t.
  do while rev
    r=''
    @20,2 say 'Apasati x pentru o optiune !' color r+/b
    @ 1,2 say 'Intrebarea nr: '+allt(str(recno()))
    @ 2,4 say Intrebare color  r+/b+
    nr=0
    k=row()+1
    do while recno()=crt
      @row()+1,20 say ' [ ] '+b.variante
      skip
      nr=nr+1
    enddo
```

Scanned with OKEN Scanner

```
    rev=.t.
    skip -nr
    do while recno()=crt
      @k,22 get o  valid o $ ' xX'
      k=k+1
      read
      r=r+allt(o)
      if uppe(o)='X'
        t=b.punctaj
        o=' '
      endif
      skip
    enddo
    if len(r)=1
      total=total+t
      rev=.f.
    else
      @ row()+1, 4 say 'Nu ati ales o singura varianta ! '
      wait wind 'Apasati o tasta pentru a reveni'
      skip -nr
      clear
      rev=.t.
    endif
  enddo
  if eof()
    wait wind 'Urmeaza interpretarea testului :'
    clear
  else
    wait wind 'Urmatoarea intrebare'
  endif
enddo
clear
sele 3
locate for total<=limsup
? interpret
wait wind 'Interpretarea  testului:  '
deacti wind w1
retu
```

33. Gestiunea unei biblioteci

Să se construiască următorul meniu:

Inscriere	Imprumut	Restituire	Carti	Iesire

pentru a realiza următoarele:

a) înscrierea unei persoane în bibliotecă;

b) inventarierea cărţilor la intrarea în bibliotecă;

Scanned with OKEN Scanner

c) împrumutul unei cărţi;
d) restituirea unei cărţi:
e) afişarea cărţilor din bibliotecă:
f) ieşire din program.

Se vor folosi următoarele fişiere de date:

CARTI.DBF

CODC	AUTOR	TITLU
C,6	C,15	C,15

CODURI.DBF

CODI	CODC	DISPON
C,9	C,6	L

PERSOANE.DBF

NUME	PRENUME	BI
C,10	C,10	C,8

IMPRUMUT.DBF

CODI	BI	DATAI
C,9	C,8	D

```
set talk off
clear
close all
use carti
use coduri in 2
use persoane in 3
use imprumut in 4
sele 2
define window w from 1,1 to 18,78
defi menu general
defi pad p1 of general prompt '\<Inscriere'
defi pad p2 of general prompt 'I\<mprumut'
defi pad p3 of general prompt '\<Restituire'
defi pad p4 of general prompt '\<Carti existente'
defi pad p5 of general prompt 'I\<esire'
defi popu po2 promp fiel a.titlu+a.autor
defi popup po1
defi bar 1 of po1 prompt '\<Persoane'
defi bar 2 of po1 prompt '\<Carti'
on pad p1 of general acti popup po1
on sele bar 1 of po1 do inscrpers
on sele bar 2 of po1 do inscrcarte
defi popu pers  promp fiel   c.nume+c.prenume
on sele popu pers do alege
on  pad p2 of general acti popu pers
on  pad p3 of general acti popu pers1
```

Scanned with OKEN Scanner

```
defi popu pers1  promp fiel    c.nume+c.prenume
on sele popu pers do alege
on sele popu pers1 do restit
on pad p4 of general acti popu po2
on sele pad p5 of general do ies
acti menu general
**********
procedure ies
deacti menu general
close all
clear
return
**********
procedure inscrpers
acti wind w
m.nume=space(10)
m.prenume=space(10)
m.bi=space(8)
@ 2,2 say 'Numele '  get m.nume valid not empty(m.nume)
@ 3,2 say 'Prenumele ' get m.prenume valid not empty(m.prenume)
@ 4,2 say 'Introdu BI ' get m.bi valid len(allt(m.bi))=8
read
sele c
locate for upper(allt(m.bi))=upper(allt(bi))
if found()
  @ 5,2 say 'Persoana  este deja inscrisa !'
  else
  append blank
  gather memvar
  @ 5,2 say 'Am facut inscrierea!'
endif
wait wind 'Apasati o tasta pentru continuare'
clear
deacti wind w
return
procedure inscrcarte
acti wind w
m.titlu=space(15)
m.autor=space(15)
@ 3,2 say 'Titlul cartii:' get m.titlu valid not empty(m.titlu)
@ 4,2 say 'Autorul     :' get m.autor valid not empty(m.titlu)
read
sele 1
locate for upper(allt(m.titlu))=upper(allt(titlu)) and;
                        upper(allt(m.autor))=uppe(allt(autor))
if found()
  ? m.titlu,titlu,m.autor,autor
  wait
```

Scanned with OKEN Scanner

```
  @5,2 say 'Acest titlu mai exista in biblioteca.'
  m.codc=codc
else
  m.codc=allt(str(recc()+1))
  appe blank
  gath memvar
endif
nrvol=0
@6,2 say 'Dati nr. de vol. ce se adauga:' get nrvol range 1,20
read
sele 2
count for codc=m.codc to volexist
for i=1 to nrvol
   appe blank
   repl codc with m.codc
   repl codi with allt(m.codc)+space(6-len(allt(m.codc)))+;
                                          allt(str(i+volexist))
   repl dispon with .t.
endfor
@ 12,2 say  'S-au  inventariat cartile'
wait wind  'Apasati o tasta pentru continuare'
clear
deacti wind w
return
**********
procedure alege
acti wind w
da='d'
do while   da='d'
  clea
  t=space(15)
  a=space(15)
  @ 2,1 say 'Titlul cartii  :' get t valid not empty(t)
  @ 3,1 say 'Autorul        :' get a valid not empty(a)
  read
  sele 1
  locate for upper(titlu)=upper(allt(t)) and upper(autor);
                                              =upper(allt(a))
  if not found()
    @ 4,1 say 'Nu exista cartea dorita'
    wait wind
  else
    @ 4,1 say 'Exista cartea  '+allt(titlu)+' de '+ allt(autor)
    sele 2
    locate for codc=a.codc and dispon
    if found()
      repl dispon with .F.
      sele 4
```

Scanned with OKEN Scanner

```
      append blank
      repl bi with c.bi , codi with b.codi , data_i with date()
      wait 'S-a imprumutat cartea cu codul '+codi
    else
      wait 'Toate vol. sunt imprumutate'
    endif
  endif
  @ 8,1 say 'Doriti alte carti ? d/n ' get da
  read
endd
clear
deacti wind w
return
**********
procedure restit
acti wind w
sele 4
loca for c.bi=bi
bi2=bi
i=0
scan for bi2=bi
  i=i+1
  sele b
  loca for codi=d.codi
  sele a
  loca for b.codc=codc
  r='d'
  @3+i,1 say 'Restitui '+allt(titlu)+' de '+allt(autor)+'?' get r
  read
  if r='d'
    sele d
    delete
    pack
    sele b
    repl dispon with .T.
    wait wind 'S-a restituit cartea'
  endif
  sele 4
ends
if i=0
  @3,1 say 'Persoana nu are carti imprumutate'
  wait wind 'Apasati o tasta ..'
endif
clear
deacti wind w
return
```

Scanned with OKEN Scanner

33. Viața în școală

Se dă baza de date `ELEVI.DBF` având articolele cu următoarea structură:

`NRM` – număr matricol;
`NUME` – nume elev;
`CLASA` – 3 caractere din care primele două semnifică anul, următorul litera clasei;
`ADR` – adresa elevului, cuprinde: strada – 10 caractere;
numărul – 3 caractere;
localitatea – 7 caractere;
`NRTEL` – număr de telefon;
`DATAN` – data nașterii;
`ABS` – număr total de absențe;
`N1, N2, N3` – note obținute la examen;
`MED` – media generală;
`ADMIS` – are valoarea `.T.` (admis, promovat) sau `.F.` (respins, nepromovat);

Se consideră respins la examen elevul care are cel puțin una din note mai mică decât 5 sau media generală mai mică decât 6.

Se cer următoarele:

1. Să se completeze câmpul `MED` cu media aritmetică a notelor `N1, N2, N3` pentru toate înregistrările.
2. Să se afișeze numele elevului cu cea mai mare medie din fiecare clasă.
3. Să se adauge o nouă înregistrare în baza de date creată prin intermediul unei machete ecran.
4. Să se listeze înregistrările pentru o clasă citită de la tastatură cu elevii a căror medie este mai mare decât 8.
5. Să se marcheze pentru ștergere înregistrările pentru care câmpul `MED` este mai mic decât 5 și să se numere înregistrările marcate.
6. Să se ordoneze înregistrările pe clase și în cadrul claselor pe medii.
7. Să se obțină lista elevilor cu medii între 9 și 10.
8. Se citesc de la tastaură un nume de elev și o clasă. Să se listeze toți elevii din școală cu medii mai mici decât media elevului respectiv.
9. Să se proiecteze o machetă care să afișeze numele elevilor, clasa și să permită introducerea absențelor și a notelor `N1, N2, N3`.
10. Să se obțină pentru fiecare clasă numărul elevilor cu medii între 5 și 6.
11. Să se afișeze numele și clasa pentru elevii admiși la examen.

Scanned with OKEN Scanner

12. Să se ordoneze prin indexare înregistrările bazei de date după data naşterii, iar pentru elevii născuți în aceeaşi dată, după nume.
13. Să se listeze numele şi clasa pentru elevii care au note mai mici decât 5 la două din probele de examen.
14. Să se listeze clasa şi adresa tuturor elevilor promovați, pe clase.
15. Să se înlocuiască în câmpul ADR strada cu o valoare citită de la tastatură pentru elevul X.
16. Să se completeze câmpul ADMIS cu valoarea .T. dacă se îndeplineşte condiția dată în enunț.
17. Să se listeze numele şi clasa elevilor care au numărul matricol egal cu numărul curent al articolului (dacă există), sau să se furnizeze un mesaj în caz contrar.
18. Să se listeze elevii admişi dintr-o clasă citită de la tastatură.
19. Să se construiască o machetă pentru modificarea numelui şi adresei elevilor dintr-o clasă citită de la tastatură.
20. Să se modifice câmpul CLASA la începerea unui nou an şcolar (absolvenții vor fi marcați pentru ştergere).
21. Să se afişeze pentru fiecare clasă numărul elevilor care nu au absențe.
22. Să se afişeze sub formă de raport situația la examen cu numele şi media elevilor.
23. Să se marcheze pentru ştergere elevii care au lipsit cel puțin la o probă de examen.
24. Să se obțină numărul total de absențe pe clase şi pe şcoală.
25. Să se afişeze numele şi clasa acelor elevi pentru care numărul zilei sau lunii de naştere coincid cu ziua sau luna curentă.
26. Să se modifice absențele interactiv, fixând coloana ABS pentru elevii unei clasei citite de la tastatură.
27. Să se listeze numele şi clasa elevilor care domiciliază într-o localitate specificată, introdusă de la tastatură.
28. Să se afişeze procentajul de elevi admişi din fiecare clasă.
29. Să se afişeze toți colegii de clasă ai elevului Ionescu.
30. Să se verifice dacă a fost completat corect câmpul ADMIS specificându-se care sunt articolele (prin numărul lor) care au fost completate eronat şi câte sunt eronate în baza de date.
31. Să se modifice toate numerele matricole conform formulei: NRM=NRM+100.
32. Să se listeze toți elevii care au vârsta peste 18 ani; în cazul în care nu există nici unul să se afişeze un mesaj.

Scanned with OKEN Scanner

33. Să se listeze elevii din 12C, începând de la primul elev promovat al acestei clase.

34. Să se înlocuiască adresa şi numărul de telefon ale elevilor tuturor claselor a 12-a prin intermediul unei machete.

35. Să se afişeze numele şi clasa elevului cu număr maxim de absenţe din şcoală şi numele clasei care are cele mai puţine absenţe.

36. Să se facă validarea verificând dacă valoarea câmpului MED e cuprinsă între 0 şi 10, marcând pentru ştergere cazurile de eroare, iar la cerere să se facă ştergerea fizică a lor.

37. Să se listeze numărul elevilor respinşi pentru fiecare clasă.

38. Să se construiască un raport cu numele şi data naşterii elevilor care au mediile între 6 şi 8.

39. Să se listeze numele şi notele elevilor din clasele care au cel puţin doi elevi admişi cu medii peste 6.

40. Să se afişeze mediile generale pe clase sub forma: CLASA MEDIA

Soluţii:

1.

```
set talk off
use elevi
scan
  replace med with (n1+n2+n3)/3
ends
sir='Lista mediilor elevilor'
clea
?space((80-len(sir))/2 )+sir
?space(8)+'Nume'+space(8)+'|Nota1|Nota2|Nota3|Media'
?replicate('-',44)
i=1
go top
do while not eof()
  if mod(i,20)=0
    wait 'Apasati o tasta!'
    clea
  endi
  ?nume+'|'+str(n1,5,2)+'|'+str(n2,5,2)+'|'+str(n3,5,2)+'|';
                                              +str(med,5,2)
  i=i+1
  skip
endd
close all
```

Scanned with OKEN Scanner

2.

```
set talk off
use elevi
sort to sortclmed on clasa,med/d
sir='Lista celor mai buni elevi pe clase'
clea
?space((80-len(sir))/2)+sir
?space((80-len(sir))/2)+replicate('-',len(sir))
?
i=1
use sortclmed
do while not eof()
  cls=clasa
  medie=med
  ?'Clasa '+clasa+':'
  ?repl('-',10)
  i=i+2
  do while clasa=cls and med=medie
    ?nume+str(med,5,2)
    i=i+1
    if i%16=0
      wait 'Apasati o tasta!'
      clea
    endi
    skip
  endd
  do while clasa=cls
    skip
  endd
endd
close all
```

3.

```
use elevi
ok=.t.
do while ok
  appe blank
  clea
  @8,15 say 'Numar matricol:' get nrm default 0
  @row()+1,15 say 'Nume:' get nume default space(20)
  @row(),col()+1 say 'Clasa:' get clasa default space(3)
  @row()+1,15 say 'Strada:' get str default space(10)
  @row(),col()+1 say 'Nr.:' get nr default space(3)
  @row(),col()+1 say 'Localitate:' get loc default space(7)
  @row()+1,15 say 'Telefon:' get nrtel default 0
  @row()+1,15 say 'Data nasterii:' get datan default {//}
  @row()+1,15 say 'Nr. absente:' get abs default 0
  @row()+1,15 say 'Nota1:' get n1 default 0
```

Scanned with OKEN Scanner

```
  @row(),col()+1 say 'Nota2:'get n2 default 0
  @row(),col()+1 say 'Nota3:' get n3 default 0
  read
  replace adr with substr(str,1,10)+space(10-len(str))+;
                substr(nr,1,3)+space(3-len(nr))+substr(loc,1,7)+;
                                                space(7-len(loc))
  replace med with (n1+n2+n3)/3
  if n1>=5 and n2>=5 and n3>=5 and med>=6
    replace admis with .t.
  else
    replace admis with .f.
  endif
  r=' '
  @15,15 say 'Mai introduceti articole?(D/N)' get r
  read
  if not r$'dD'
    ok=.f.
  endi
endd
close all
```

4.

```
set talk off
use elevi
clea
@10,10 say 'Dati clasa:' get cls default space(3)
read
disp  nume,med for clasa=cls and med>8 off
close all
```

5.

```
set talk off
use elevi
nr=0              &&numara inregistrarile marcate pentru stergere
scan for med<5
  dele
  nr=nr+1
ends
clea
?'Au fost sterse logic'+str(nr,4)+' inregistrari'
close all
&& Se poate lucra si cu if daca se renunta la conditia din scan
```

6.

```
set talk off
use elevi
index on clasa+str(med,5,2) to ind
clea
```

Scanned with OKEN Scanner

```
sir='Lista elevilor pe clase in ordinea mediilor'
?space((80-len(sir))/2)+sir
?
?space(8)+'NUME'+space(12)+'CLASA MEDIA'
i=3
use elevi index ind
do while not eof()
  ?str(i-2,2)+'. '+ nume+' '+clasa+'  '+str(med,5,2)
  i=i+1
  if i%20=0
    wait
    @4,1 clea to 23,80
  endi
  skip
endd
close all
```

7.

```
set talk off
use elevi
sort to sortmed on med/d
use sortmed
clea
sir='Lista elevilor cu medii intre 9 si 10'
?space((80-len(sir))/2)+sir
i=2
scan while med>=9
  ?str(i-1,2)+'. '+nume+ clasa+str(med,7,2)
  i=i+1
  if i%20=0
    wait
    @3,1 clea to 22,80
  endi
ends
if i=2
  ?'Nu sunt copii cu media peste 9'
endi
close all
```

8.

```
set talk off
clear
@2,2 say 'Numele:' get x default space(20)
@3,2 say 'Clasa:' get y default space(3)
read
use elevi
locate for nume=x and clasa=y
```

Scanned with OKEN Scanner

```
if found()
  medx=med
  clea
  disp all nume,med for med<medx
else
  clea
  @10,10 say ' Elevul '+x+;
             ' nu este in baza de date sau nu este in clasa '+y
endi
close all
```

9.

```
set talk off
use elevi
do while not eof()
  clea
  @10,10 to 16,52 double
  @11,11 say 'NUME:'+nume
  @row(),col()+1 say 'CLASA:'+clasa
  @row()+1,11 say 'Nr. absente:' get abs default 0
  @row()+1,11 say 'Nota 1:' get n1 default 0
  @row(),col()+1 say 'Nota 2:' get n2 default 0
  @row(),col()+1 say 'Nota 3:' get n3 default 0
  read
  r=' '
  @15,11 say 'Mai modificati?(d/n)?' get r
  read
  if r$'nN'
    exit
  else
    skip
  endi
endd
close all
```

10.

```
set talk off
use elevi
sort to sortcl on clasa
use sortcl
clea
sir='Lista numarului de elevi cu medii intre 5 si 6 pe clase'
?space((80-len(sir))/2)+sir
?
i=2
do while not eof()
  nr=0
  cls=clasa
```

Scanned with OKEN Scanner

```
  if i%20=0
    wait
    @3,0 clea to 23,80
  endi
  ?'Clasa '+clasa
  i=i+2
  do while clasa=cls
    if med>=5 and med<=6
      nr=nr+1
    endi
    skip
  endd
  ?iif(nr=0,'Nu sunt!','Sunt'+str(nr,3))
  ?
  i=i+1
endd
close all
```

11.

```
set talk off
use elevi
sort to sortcl on clasa
use sortcl
replace all admis with .t. for n1>=5 and n2>=5 and n3>=5;
                                                    and med>=6
clea
?'Lista elevilor admisi la examen'
?
go top
i=2
j=0
do while not eof()
  if admis
    if i=23
      wait
      @3,0 clea to 24,80
      i=2
    endi
    j=j+1
   ?str(j,2)+'. '+nume+clasa
    i=i+1
  endi
  skip
endd
close all
```

Scanned with OKEN Scanner

12.
```
set talk off
use elevi
index on str(year(datan),4)+str(month(datan),2)+;
                                    str(day(datan),2) +nume to ind
use elevi index ind
disp all nume,datan off
close all
```

13.
```
set talk off
use elevi
sir='Elevii cu doua note sub 5 la examen'
clea
?space((80-len(sir))/2)+sir
?
i=2
do while not eof()
  if (n1<5 and n2<5) or (n1<5 and n3<5) or (n2<5 and n3<5)
    if i%20=0
      wait
      @3,0 clea to 23,80
    endi
    i=i+1
    ?str(i-2,2)+'. '+nume+clasa
  endi
  skip
endd
if i=2
  ?'Nu exista elevi cu doua note sub 5'
endi
close all
```

14.
```
set talk off
clea
use elevi
index on clasa tag cl for admis
do while !eof()
  c=clasa
  ?"promovatii din clasa:",c
  disp nume,adr while clasa=c
endd
close all
```

Scanned with OKEN Scanner

15.

```
set talk off
clea
x=space(20)
strada=space(10)
@2,2 say 'Nume elev:' get x
@3,2 say 'Noua strada:' get strada
read
use elevi
locate for nume=x
if found()
  repl adr with stuff(adr,1,10,strada)
  clea
  disp nume,adr
else
  clea
  ?'Nu exista elev cu numele '+x+' in fisier'
endi
close all
```

16.

```
set talk off
use elevi
scan
  if (n1>=5)and(n2>=5)and(n3>=5)and med>=6
    replace admis with .t.
  endi
ends
go top
disp all nume,n1,n2,n3,med,admis off
close all
```

17.

```
set talk off
use elevi
OK=.f.
clea
scan
  if nrm=recno()
    ? nume+clasa
    OK=.t.
  endi
ends
if not OK
  ? 'Nu exista elevi cu nr. matricol = nr. curent al
articolului'
endi
close  all
```

Scanned with OKEN Scanner

18.
```
set talk off
clea
accept 'Clasa:' to cls
clea
sir='Elevii admisi din clasa ' + cls
? space((80-len(sir))/2)+sir
?
use elevi
sort to sortcl on clasa
use sortcl
locate for clasa = cls
do while clasa=cls
  i=i+1
  ?str(i,2) + '. ' + nume
  skip
endd
close all
```

19.
```
set talk off
use elevi
sort to sortcl on clasa
use sortcl
@2,2 say 'clasa dorita:' get c defa space(3)
read
locate for clasa = c
clea
@9,19 to 12,48 double
do while clasa=c
  @10,20 clea to 11,47
  @10,20 say 'Nume:' get nume
  @11,20 say 'Adresa:' get adr
  read
  skip
endd
close all
```

20.
```
set talk off
use elevi
dele for left(clasa,2)='12'
do while not eof()
  if val(left(clasa,2))<>12
    replace clasa with stuff(clasa,1,2,str(val(left(clasa,2))+1,2))
  endi
  skip
endd
```

Scanned with OKEN Scanner

```
go top
disp all
close all
```

21.

```
set talk off
use elevi
sort to sortcl on clasa
use sortcl
clea
sir='Numarul elevilor fara absente'
?space((80-len(sir))/2)+sir
?
i=2
do while not eof()
  cls=clasa
  j=0
  do while clasa=cls
    if abs=0
      j=j+1
    endi
    skip
  endd
  if i=23
    wait
    @3,0 clea to 24,80
    i=2
  endi
  ?'In clasa '+cls
  if j>1 or j=0
    ??iif(j=0,'Nu sunt',' sunt'+str(j,3)+' elevi')
  else
    ??' este 1 elev'
  endi
endd
close all
```

22.

```
set talk off
use elevi
replace all med with (n1+n2+n3)/3
go top
replace all admis with .t. for n1>=5 and n2>=5 and n3>=5 and
med>=6
go top
sort to sortmenu on med/d,nume
use sortmenu
sir='Rezultatele la examen'
```

Scanned with OKEN Scanner

```
clea
?space((44-len(sir))/2)+sir
?repl('-',44)
?'|'+'NR.CRT|'+space(8)+'NUME'+space(8)+'|MEDIA|REZULTAT|'
?repl('-',44)
i=5
j=0
do while not eof()
  if i=23
    wait
    @6,0 clea to 24,80
    i=5
  endi
  j=j+1
  ?'|'+str(j,5)+'.|'+nume+'|'+str(med,5,2)+'|'
  if admis=.t.
    ??'  admis |'
  else
    ??' respins|'
  endi
  i=i+1
  skip
endd
?repl('-',44)
?'Rezultatele au fost afisate astazi: '+str(day(date()),2)+;
cmonth(date())+str(year(date()),4)+'.'
wait
close all
```

23.

```
use elevi
scan
  if n1=0 or n2=0 or n3=0
    dele
  endi
ends
disp all nume,n1,n2,n3
close all
```

24.

```
set talk off
use elevi
sort to sortcl on clasa
use sortcl
total to totabs on clasa fields abs
sir='Totalul absentelor pe clase'
clea
?space((80-len(sir))/2)+sir
```

Scanned with OKEN Scanner

```
i=2
use totabs
do while not eof()
  if i=23
    wait
    @3,0 clea to 24,80
    i=2
  endi
  ?'Clasa '+clasa+repl('.',16)+str(abs,3)
  i=i+1
  skip
endd
go top
sum abs to a
?'Total absente pe scoala:'+str(a,4)
close all
```

25.

```
set talk off
use elevi
clea
ok=.f.
i=1
scan
  if i=23
    wait
    @1,0 clea to 24,80
    i=1
  endi
  if day(datan)=day(date()) or month(datan)=month(date())
    ?nume,clasa,datan
    i=i+1
  endi
ends
if i=1
  ?'Nu sunt elevi nascuti in ziua sau luna curenta'
endi
close all
```

26.

```
set talk off
@2,2 say 'Clasa:' get c defa '   '
read
use elevi
clea
wait 'Terminarea modificarilor ulterioare se va face prin;
                                                    <ctrl>+<End>'
brow  for clasa=c fields nume:r,abs
close all
```

Scanned with OKEN Scanner

27.

```
set talk off
clea
accept 'Localitatea:' to loc
sir='Elevii care locuiesc in localitatea '+loc
?space((40-len(sir))/2)+sir
i=2
j=0
use elevi
do while not eof()
  if i=23
    wait
    @3,0 clea to 24,80
    i=2
  endi
  if right(adr,7)=loc
    j=j+1
    ?str(j,2)+'. '+nume+clasa
    i=i+1
  endi
  skip
endd
close all
```

28.

```
set talk off
use elevi
sort to sortcl on clasa
use sortcl
nre=0
nra=0
sir='Procentajul elevilor admisi pe clase'
clea
?space((80-len(sir))/2)+sir
i=2
do while not eof()
  if i=23
    wait
    @3,0 clea to 24,80
    i=2
  endi
  cls=clasa
  do while clasa=cls
    if admis=.t.
      nra=nra+1
    endi
    nre=nre+1
    skip
  endd
```

Scanned with OKEN Scanner

```
  ?'Clasa:'+cls+': '
  ??str(nra*100/nre,6,2)+'%'
endd
close all
```

29.

```
set talk off
use elevi
locate for upper(left(nume,7))='IONESCU'
if found()
  cls=clasa
  n=nume
  clea
  go top
  scan for clasa=cls
    if nume<>n
      ?nume
    endi
  ends
else
  clea
  ?'Nu exista elevul Ionescu in baza de date'
endi
close all
```

30.

```
set talk off
use elevi
nerr=0
clea
?'Articole completate eronat '
do while not eof()
  if admis and (n1<5 or n2<5 or n3<5 or med<6)
    ??recno()+' '
    nerr=nerr+1
  endi
  if not admis and n1>=5 and n2>=5 and n3>=5 and med>=6
    ??recno()+' '
    nerr=nerr+1
  endi
  skip
endd
if nerr=0
  ??'nu sint in fisier!'
else
  ?'In baza sint'+str(nerr,4)+' completate eronat'
endi
close all
```

Scanned with OKEN Scanner

31.

```
set talk off
use elevi
repl all nrm with nrm+100
disp all nume,nrm
close all
```

32.

```
set talk off
use elevi
clea
?'Elevi care au implinit 18 ani'
i=2
j=0
do while not eof()
  if i=24
    wait
    @3,0 clea to 24,80
    i=2
  endi
  v=int((date()-datan)/365.25)
  if v>=18
    ?nume,clasa,datan
    i=i+1
    j=j+1
  endi
  skip
enddo
if j=0
  ??' nu  exista in fisier.'
endi
close all
```

33.

```
set talk off
use elevi
sort to sortclmed on clasa,med/d
use sortclmed
clea
?'Elevii admisi din clasa 12C'
i=2
j=0
locate for clasa='12C'
do while not eof() and (clasa='12C')and (admis=.t.)
  if i=23
    wait
    @3,0 clea to 24,80
    i=2
  endi
```

Scanned with OKEN Scanner

```
  j=j+1
  ?str(j,2)+'. '+nume
  skip
endd
clear
close all
```

34.

```
set talk off
use elevi
sort to sortcl on clasa
use sortcl
scan for left(clasa,2)='12'
  clea
  @10,10 to 14,50 double
  @11,11 say 'NUME:'+nume
  @11,col()+1 say 'CLASA:'+clasa
  @12,11 say 'ADRESA:' get adr
  @13,11 say 'TELEFON:' get nrtel pict '9999999'
  read
ends
close all
```

35.

```
set talk off
use elevi
sort to sortabs on abs/d
use sortabs
a=abs
clea
?'Elevii cu cele mai multe absente'
do while abs=a
  ?nume,clasa
  skip
endd
go top
sort to sortcl on clasa
use sortcl
total to totabs on clasa
use totabs
sort to sorttot on abs
use sorttot
a=abs
wait
clea
?'Clasele cu cele mai putine absente'
list clasa for abs=a
close all
```

Scanned with OKEN Scanner

36.

```
set talk off
use elevi
i=0
do while not eof()
  if not between(med,0,10)
    dele
    ?recno(),nume,med
    i=i+1
  endi
  skip
endd
if i=0
  clea
  @10,20 say 'Nu au fost sterse logic articole!'
else
  clea
  @10,20 say 'Sint marcate pentru stergere'+str(i,3)+'articole'
  @11,20 say 'Doriti stergere fizica?' get rasp default 'N'
  read
  if upper(rasp)='D'
    pack
  endi
endi
close all
```

37.

```
set talk off
use elevi
sort to sortcl on clasa
use sortcl
clea
sir='Numarul elevilor respinsi din fiecare clasa'
?space((80-len(sir))/2)+sir
i=2
do while not eof()
  j=0
  if i=23
    wait
    @3,0 clea to 24,80
    i=2
  endi
  ?'Clasa:'+clasa
  cls=clasa
  do while clasa=cls
    if admis=.f.
      j=j+1
    endif
```

Scanned with OKEN Scanner

```
    skip
  endd
  ??str(j,4)
  i=i+2
endd
close all
```

38.

```
set talk off
use elevi
sir='Elevii cu medii intre 6 si 8'
clea
?space((44-len(sir))/2)+sir
?repl('-',44)
?'|NR.CRT.|'+space(8)+'NUME'+space(8)+'|DATA NASTERII|'
?repl('-',44)
i=0
j=5
scan for between(med,6,8)
  if j=23
    wait
    @6,0 clea to 24,80
    i=5
  endi
  i=i+1
  ?'|'+str(i,6)+'.|'+nume+'|'+space(3)+dtoc(datan)+'   |'
  j=j+1
ends
?repl('-',44)
close all
```

39.

```
set talk off
use elevi
sort to sortcl on clasa
use sortcl
i=0
clea
do while not eof()
  cls=clasa
  na=recno()
  ne=0
  do while clasa=cls
    if med>6
      ne=ne+1
    endi
    skip
  endd
```

Scanned with OKEN Scanner

```
  if ne>=2
    go na
    ?
    do while clasa=cls
      if i=24
        wait
        clea
        i=0
      endi
      ?nume,med,clasa
      i=i+2
      skip
    endd
  endi
endd
close all
```

40.

```
set talk off
use elevi
sort to sortcl on clasa
use sortcl
clea
?
?space(10)+repl('_',13)
?space(10)+'|Clasa|Media|'
?space(10)+repl('-',13)
i=4
do while not eof()
  cls=clasa
  ne=0
  mg=0
  do while clasa=cls
    ne=ne+1
    mg=mg+med
    skip
  endd
  if i=24
    wait
    @4,0 clea to 24,80
    i=4
  endi
  ?space(10)+'| '+cls+' |'+str(mg/ne,5,2)+'|'
  ?space(10)+repl('_',13)
  i=i+2
endd
close all
```

Scanned with OKEN Scanner

III. MEDIUL WINDOWS – MICROSOFT WORD

1. Grupuri de aplicații

a) Să se creeze un grup de aplicații în sistemul Windows care are subicon numele BorlandC.
b) Să i se adauge acestui grup aplicația G:\Borlandc\Bin\Bcw.exe.
c) Să se mute în acest grup aplicația Write existentă în Accessories.
d) Să se afişeze pentru directorul C:\Windows în fereastra director numai conținutul directorului curent în care să fie vizibile numai directoarele şi fişierele executabile (cu extensia `.exe`, `.com`, `.bat` etc.) cu data ultimei actualizări.
e) Să se ordoneze aceste fişiere descrescător după dată, apoi să se şteargă (utilizând comanda DELETE o singură dată) toate fişierele executabile, modificate anterior lunii curente.
Atenție!!!
Toate aceste modificări nu trebuie salvate pentru următoarea sesiune Windows!

Soluție:

a) În Program Manager: meniul File, comanda New
– în fereastra New Program Object se alege butonul de opțiune Program Group;
– în fereastra Program Group Properties, în zona Description, se introduce textul "BorlandC".
b) Cu fereastra grupului BorlandC deschisă (ca urmare a creării lui), se alege comanda New din meniul File;
– în fereastra New Program Object se alege butonul de opțiune Program Item;
– în fereastra Program Item Properties se completează un nume pentru aplicația nouă în zona Description, iar în zona Command Line se scrie:
`"G:\Borlandc\Bin\Bc.exe"` sau se caută aplicația pe disc folosind butonul Browse.
c) Se deschide grupul Accessories, astfel încât cele două ferestre de grup să fie ambele vizibile pe ecran (folosind, la nevoie, comanda Tile din meniul Window);
– se selectează aplicația Write; se foloseşte comanda Move din meniul File; în fereastra Move Program Item se completează, în zona To Group, numele grupului nou creat (BorlandC) sau se alege din listă.
Alternativă: mutare cu mouse-ul prin dragging în grupul BorlandC.
d) În File Manager:
– se selectează directorul C:\Windows; din meniul View, se alege comanda Directory Only, apoi comanda By File Type; în fereastra By File Type, în zona File Type, se selectează doar căsuța de opțiune Programs.

Scanned with OKEN Scanner

e) din meniul View, comanda Sort By Date; apoi se selectează toate fişierele executabile anterioare lunii curente, folosind mouse-ul (click pe primul fişier, Shift + click pe ultimul) sau tastatura (Shift + săgeţi);
– meniul File, comanda Delete.

2. În File Manager

a) Să se afişeze într-o fereastră director conţinutul directorului C:\WINDOWS, astfel încât pentru fiecare fişier din acest director să fie vizibile toate proprietăţile (*details*).
b) Să se selecteze toate fişierele având extensia .com şi să li se atribuie – printr-o singură comandă – atributul Read-Only.
c) Să se afişeze, detaliată până la ultimul nivel, întreaga structură de directoare de pe discul C.
Atenţie!!!
Toate aceste modificări nu trebuie salvate pentru următoarea sesiune Windows!

Soluţie:

a) meniul View, comenzile Directory Only, All File Details.
b) meniul File, comenzile Select Files, Properties.
c) meniul Tree, comanda Expand All.
Pentru a nu salva modificările făcute: în meniul Options, se deselectează opţiunea Save Settings On Exit.

3. În Control Panel

a) Să se creeze un nou *pattern* prin modificarea *patternului* Diamonds şi să se afişeze un *wallpaper* – de exemplu: Flying Windows – centrat.
b) Să se creeze o nouă structură de culori în care:
b1) bara de titlu a ferestrei active va fi roşie;
b2) bara de titlu a ferestrei inactive va fi bleu;
b3) fondul ferestrei va fi albastru;
b4) comanda selectată va fi maro;
b5) umbra butoanelor va fi galbenă.
Atenţie!!!
Toate aceste modificări nu trebuie salvate pentru următoarea sesiune Windows!

Soluţie:
a) Din aplicaţia Desktop, în zona Pattern, butonul Edit Pattern; după schimbarea numelui, butonul Change.
b) Din aplicaţia Color, în zona Color Schemes, se alege culoarea cerută pentru fiecare componentă a ecranului, se foloseşte butonul Save Scheme.

Scanned with OKEN Scanner

4. Modificarea unui document Word (1)

În directorul C:\ATESTAT există un document ATEST1.DOC şi un desen SALI.BMP.

Să se efectueze modificările precizate mai jos asupra fişierului ATEST1.DOC dând naştere documentului ATEST2.DOC, totodată păstrându-se şi documentul vechi cu denumirea sa.

a) Textul din document se va formata în felul următor:
- caracterele vor fi scrise cu fontul Times New Roman CE de dimensiune 12 pts.;
- tabulatorul la stânga se fixează la 1 cm;
- textul se aliniază cu fixarea marginilor (aliniat şi la stânga şi la dreapta);
- titlul se centrează şi se modifică la majuscule, boldate de 17 pts.

b) La sfârşitul textului se importă desenul din fişierul menţionat, centrat, mărit astfel încât să ocupe o pagină întreagă (nu se modifică setarea paginii fişierului ATEST1.DOC).

c) După desen se va insera un tabel care va fi completat cu sălile din figura importată şi cu lista candidaţilor de la sfârşitul documentului ATEST1.DOC. Cunoscând din desen numărul locurilor existente în fiecare sală, se va crea un tabel (centrat pe pagină) având următoarea structură:

Nr.crt	Nume şi prenume	Clasa	Sala

Soluţie:

a) – meniul Format, comanda Font
- meniul Format, comanda Tabs
- meniul File, comanda Page Setup
- se selectează titlul, din bara de formatare se alege butonul Center; din meniul Format, comanda Font (All Caps) şi butonul Bold de pe bara de formatare.

b) Meniul Insert, comanda Picture; după inserare, se selectează pentru editare desenul, se mută şi se redimensionează cu ajutorul mouse-ului.

c) Meniul Table, comanda Insert Table; pentru formatarea tabelului, se selectează (meniul Table, comanda Select Table), apoi se foloseşte din meniul Format comanda Borders And Shading.

Pentru centrarea tabelului, se foloseşte meniul Table, comanda Cell Height And Width, eticheta Row, butonul de opţiune Center din zona Alignment.

Scanned with OKEN Scanner

5. Modificarea unui document Word (2)

În directorul C:\ATESTAT există un document ATEST3.DOC.

a) Să se efectueze modificările precizate mai jos asupra fişierului ATEST3.DOC dând naştere documentului ATEST4.DOC, totodată păstrându-se şi documentul vechi cu denumirea sa.

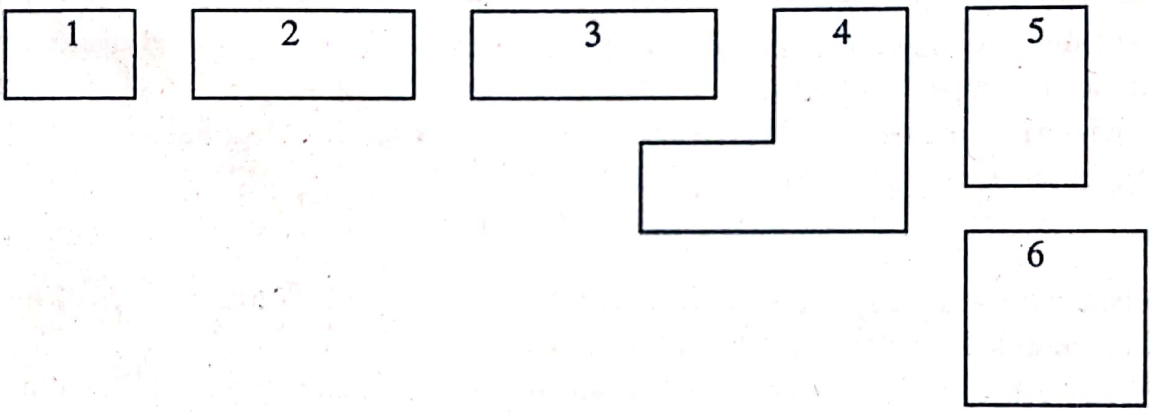
Titlul documentului se va încadra cu umbre.

b) Să se adauge documentului un *header* de forma:

Titlul documentului nr. pagină

pentru fiecare pagină impară şi invers pentru fiecare pagină pară. Prima pagină nu are *header* doar număr de pagină.

c) Să se introducă în document, imediat după subtitlul "Harta sălilor", desenul:



Soluţie:

a) Se selectează titlul, se alege comanda Borders And Shading din meniul Format.

b) Meniul View, comanda Header And Footer; pentru inserarea numărului paginii se foloseşte butonul adecvat din bara de editare a headerului; pentru afişarea diferenţiată pe pagini pare şi impare se foloseşte butonul Page Setup, apoi se selectează zonele Different Odd And Even şi Different First Page. Se inserează în Header un tabel, a cărui linie de jos se marchează ca desenată, în coloana din stânga se introduce textul dorit (cu aliniere la stânga), în coloana din dreapta numărul paginii (cu aliniere la dreapta), respectiv invers pentru paginile pare.

c) După poziţionarea specificată în cerinţă, se folosesc butoanele barei de desenare, care devine vizibilă pe ecran folosind din meniul View comanda Toolbars, selectând căsuţa Drawing.

Se pot utiliza butoanele Rectangle şi Reshape pentru a desena şi butonul Callout pentru a insera cifre în interiorul poligoanelor.

Scanned with OKEN Scanner

6. *Modificarea unui document Word (3)*

În directorul C:\ATESTAT există două documente: ATEST5.DOC şi LISTA.DOC.

Să se efectueze modificările precizate mai jos asupra fişierului ATEST5.DOC dând naştere documentului ATEST6.DOC, totodată păstrându-se şi documentul vechi cu denumirea sa.

a) Să se mărească dimensiunile titlului la o dimensiune maximă pentru care acest titlu apare pe o singură linie.

b) Să se formateze primul paragraf pe 2 coloane.

c) Să se importe fişierul LISTA.DOC şi să se formateze conţinutul acestuia în tabel având următoarea structură:

Nr. crt.	Nume şi prenume

– în tabel Numele se va ordona alfabetic (prin comandă);
– Nr. crt. se va completa automat.

d) Să se adauge o notă de subsol la cuvântul "header" din text;

e) Să se schimbe tot ce este bold în text în bold italic.

Soluţie:

a) În documentul ATEST5.DOC se selectează titlul, apoi se măreşte folosind zona Font Size de pe bara de formatare, astfel încât să nu depăşească o linie.

b) Se selectează primul paragraf, apoi se alege din meniul Format comanda Columns, se fixează numărul de coloane la 2.

c) Meniul Insert, comanda File; se specifică fişierul LISTA.DOC. După inserare, se selectează conţinutul nou adus, se alege din meniul Table comanda Convert Text To Table. Tabelului astfel obţinut i se adaugă coloana pentru **Nr. crt.** şi capul tabelului, care se şi completează.

Sortarea se face folosind comanda Sort din meniul Table, iar completarea numărului curent prin selectarea primei coloane şi utilizarea butonului Numbering de pe bara de formatare.

d) După poziţionarea la sfârşitul cuvântului specificat, se alege din meniul Insert comanda Footnote, după care se permite editarea notei de subsol.

e) Meniul Edit, comanda Find, nu se specifică text de căutare, ci numai formatarea Bold. Se utilizează butonul Replace, nu se specifică un text pentru înlocuire, ci formatarea Bold Italic.

Se face înlocuire în întregul document, specificând în zona Search domeniul All.

Scanned with OKEN Scanner

7. Formatare text (portrait)

Preluați din Help-ul Windows 20 de rânduri de text și așezați-le într-o pagină de Word format A4 pe lungime cu margini egale peste tot (2 cm). Formatați textul astfel:
– aliniere și la stânga și la dreapta;
– distanța dintre rânduri Single;
– paragrafe la 1.5 cm;
– corp de literă 12, Arial, italic.

Soluție:

Preluarea textului se face deschizându-se simultan Help și MsWord. Se selectează porțiunea de text dorită, se copiază în Clipboard, apoi se "aduce" în MsWord cu comanda Paste. Procedeul rămâne valabil pentru oricare din problemele propuse în care se solicită preluarea unui text din altă aplicație. Se selectează apoi Page Setup din meniul File, și se stabilesc setările corespunzătoare.

Din meniul Format se stabilesc opțiunile pentru indentări, spații între rânduri, aliniere Justified, corpul și tipul de literă.

8. Formatare text (landscape)

Preluați un text oarecare de pe calculator (minim 30 rânduri) și poziționați-l într-o pagină Word format A4 pe lățime. Textul va avea umătorul format:
– 2 coloane;
– aliniere Justify (la ambele margini);
– distanța dintre rânduri Double;
– începutul de paragrafe la 2 cm de margine.

Soluție:

Noutăți față de problema anterioară: se selectează textul ce va fi scris pe coloane, eventual tot (Ctrl+A) apoi din meniul Format se alege Columns și se stabilește numărul acestora, distanța dintre ele etc.

9. Tabel (1)

Într-o pagină de Word, format A4, portrait, realizați un tabel intitulat "Tabel statistic" în care introduceți numele a 10 persoane,vârsta, înălțimea, greutatea, marca (un număr de 3 cifre). După introducere, aranjați persoanele după criteriile vârstă și înălțime. Tabelul realizat va avea la început o coloană cu număr curent generat automat, margini duble și liniile din interior simple.

Soluție:

Inserarea tabelului în pagină se va face cu opțiunea Insert Table unde se stabilesc caracteristicile tabelului. Din meniul Format / Bullets and Numbering se alege număra-

Scanned with OKEN Scanner

rea automată pe coloana selectată (cea cu numărul curent). După introducerea datelor se selectează întreg tabelul în vederea pregătirii sortării după criteriile date. Se selectează Sort din meniul Table, stabilindu-se criteriile de sortare.

10. Formatare text (3)

Preluați o secvență de text din Help-ul Word. Formatați acest text astfel:
– pagină A4 portrait toate marginile de 2cm;
– titlul "ÎNCERCARE", corp 32 TimesNewRoman bold, subliniat cu două linii;
– textul pe două coloane cu linie despărțitoare între ele;
– 3 zone de text, a câte 5 rânduri, vor fi marcate cu fond gri.

Soluție:

Preluarea textului se face ca în problemele anterioare. Titlul va fi dimensionat din opțiunea Format / Font. Se va selecta zona de text care se scrie pe coloane, apoi se va realiza operația de aranjare corespunzătoare din Format / Columns. Pentru marcarea zonelor de text, se folosește opțiunea Borders and Shading, nu înainte de a se selecta zona dorită.

11. Configurarea MsWord

Pregătiți pentru scriere editorul Word, astfel încât pagina de lucru să fie permanent (la orice nouă sesiune) A4, marginile sus, jos: 2cm, dreapta 2cm, stânga 3cm, fontul TimesNewRoman 12, indentarea de paragraf la 1.5 cm.

Preluați un text oarecare în acest format, şi aranjați-l la două rânduri.

Soluție:

Noutatea față de cerințele anterioare este permanentizarea formatului dat, lucru care se realizează cu opțiunea Default în File / Page Setup.

12. "Calculator"

Scrieți 10 rânduri de text în care să folosiți cuvântul "calculator" de 7 ori. Faceți înlocuirea automată a cuvântului "calculator" cu "tractor", asfel ca acesta din urmă să folosească alt tip de caracter decât cel inițial. Primului cuvânt "tractor" faceți-i o adnotare în subsolul paginii: "era vorba de calculator".

Soluție:

Problema apelează la funcția Replace a editorului care se află în meniul Edit. Se selectează textul în care se face înlocuirea, apoi Replace unde se transmite editorului grupul de litere ce va fi înlocuit. Pentru nota de subsol, se poziționează cursorul la sfârşitul cuvântului la care se face referire, apoi se selectează din Insert, Footnote şi se scrie nota dorită.

Scanned with OKEN Scanner

13. Pliant

Realizați o machetă de pliant:
– pagină A4, pe lățime, marginile 1.5 peste tot;
– 3 coloane de text, la un rând cu linie despărțitoare între ele;
– 3 desene, câte unul pe fiecare coloană.

Preluați fragmente de text de oriunde de pe hard-disc (help-uri, documentații), iar desene din bitmap-urile Windows.

Soluție:

Formatarea coloanelor este cunoscută din rezolvările anterioare. Desenele se inserează cu opțiunea Insert Picture sau cu Copy şi Paste după ce s-a deschis o aplicație grafică şi se copiază în Clipboard obiectul dorit. Se poziționează desenul direct în interiorul textului, reducându-se corespunzător, sau într-un frame, după ce s-a apelat funcția Insert Frame.

14. Formatare text (4)

Preluați trei sau mai multe pagini de text în Word. Întregul text va fi plasat pe pagini de format A4 cu marginile: stânga 3cm, dreapta 2cm, sus, jos câte 1.5cm. Numerotați paginile şi puneți câte una/două note de subsol la fiecare pagină. Faceți în aşa fel încât pe fiecare pagină să apară automat, în partea de sus a acesteía, numele vostru.

Soluție:

Noutatea față de problemele anterioare este numerotarea paginilor, care se realizează cu funcția Insert Page Numbers şi introducerea unui header. Din meniul View se alege Header and Footer. Fereastra deschisă permite scrierea în partea de sus sau în cea de jos a paginii a unui text scurt care va rămâne pe toate paginile documentului.

15. Formatare text (5)

Pregătiți pentru tipărire un text pe care l-ați preluat din ASCII sau Notepad (minimum 30 de rânduri). Foaia de lucru să fie format Letter, marginile peste tot 1", font Currier 12, la două rânduri, paragrafele vor fi numerotate automat cu 1.1, 1.2, 1.3... Se va face alinierea textului la ambele margini. Tipărirea se realizează schimbând manual fiecare pagină.

Soluție:

Paragrafele se numerotează selectând textul şi alegând opțiunea Bullets and Numbering, unde se va stabili formatul corespunzător al numerotării. Când se selectează Print, se va alege din fereastra corespunzătoare imprimării Options, Manual feed.

Scanned with OKEN Scanner

16. Tabel (2)

Realizaţi o pagină în care să existe un tabel cuprinzând date statistice despre inflaţie începând cu 1992-34%, 1993-23%, 1994-45%, 1995-20%, 1996-24%, 1997-34%. Inseraţi în această pagină un grafic care să ilustreze evoluţia inflaţiei şi câteva rânduri de comentarii scrise dedesubt, pe trei coloane.

Soluţie:

Tabelul va fi realizat ca în problemele anterioare. Acesta poate fi preluat într-o foaie Excell şi i se poate ataşa un grafic care se copiază şi se transportă în Word. Tabelul poate fi realizat în Excell şi transportat în Word. Tot în Excell se poate ataşa graficul corespunzător tabelului.

17. Tabel (3)

Să se realizeze un tabel de date statistice având coloanele: C1, C2, C3, MEDIA=(C1 +C2+C3) / 3, MEDIA_GEOM=max(C1, C2, C3), C1, C2, C3 fiind numere nenegative, MEDIA şi MEDIA_GEOM, fiind câmpuri calculate. Tabelul realizat va purta titlul "Tabel Statistic", va fi bordat de o linie dublă şi va cuprinde şi totalurile pe verticală pe fiecare coloană. Se vor introduce 15 înregistrări.

Soluţie:

Se va insera tabelul folosind Table / Insert Table, pentru bordare se va folosi Borders and Shanding sau icon-ul corespunzător. Câmpurile calculate: MEDIA şi MEDIA_GEOM se vor realiza cu Table / Formula înscriindu-se în fereastra corespunzătoare =AVERRAGE() respectiv, =MAX().

Pentru totalurile pe verticală se va folosi formula =SUM(above).

18. Inserare de desene

Să se introducă într-o lucrare ştiinţifică două desene în interiorul unui text astfel încât textul să încadreze desenele respective. Realizaţi un astfel de exemplu pe un text oarecare de două pagini preluate din Help-ul Windows. Un desen va reprezenta un triunghi cu notaţiile corespunzătoare, iar celălalt desen va fi un grafic "pie" realizat în Excell.

Soluţie:

Se transportă textul respectiv folosind facilitatea OLE. Se inserează frame în zonele în care se doreşte introducerea desenelor. Odată realizate acestea, se transportă cu Copy şi Paste.

Scanned with OKEN Scanner

IV. SISTEMUL DE OPERARE DOS

1. Comenzi DOS

a) Utilizând comanda `HELP` să se afle sintaxa următoarelor comenzi DOS:

```
CD              FORMAT
DISKCOPY        COPY CON
COPY            MD
CLS             RD
DEL             RENAME
DIR             PRINT
TYPE
```

Să se depună rezultatul în fişierul `HELP.TXT`.

b) Folosind comanda `MD<nume_dir>` să se creeze următoarea structură de directoare:

```
... TEST1...SDIR1
                ...SDIR2
... TEST2...SDIR1
                ...SDIR2
                ...SDIR3
```

Să se realizeze ştergerea subdirectorului `SDIR3` aflat în subdirectorul `TEST2`, folosind comenzile `CD, RD`.

Observaţiile legate de modul de lucru se vor depune în fişierul `OBSERV.TXT` creat cu comanda `COPY CON` (a cărui sintaxă o avem în `HELP.TXT`). Editarea se încheie cu CTRL+Z.

c) Fiind în directorul rădăcină, folosind comanda `TREE` redirectată, creaţi fişierul `TREE.TXT` care va permite vizualizarea structurii de directoare. Realizaţi apoi ştergerea subdirectorului `SDIR1` din `TEST2` şi salvaţi noua arborescenţă de directoare în fişierul `TREE2.TXT`.

d) Vizualizaţi lista fişierelor din directorul `C:\ATESTAT`, folosind comanda `DIR` cu cel puţin 2 parametri diferiţi.

Soluţie:

a)

```
@echo off
echo comanda echo off inhiba afisare mesajelor pe ecran
echo sterg vechiul fisier sint.txt
del sint.txt
echo apelez help <comanda> pt fiecare din elementele din lista
echo lui for
echo rezultatul comenzii il redirectez intr-un fisier sint.txt
echo prin >> se adauga la continutul vechi al fisierului
echo rezultatul curent
```

Scanned with OKEN Scanner

```
for %%v in (cd, diskcopy, copy, cls) do help %%v >> sint.txt
for %%v in (format, copy con ,md, rd) do help %%v >> sint.txt
for %%v in (rename, print, dir, type, del) do help %%v >>
sint.txt
echo tiparesc continutul fisierului pagina cu pagina
type sint.txt | more
```

b) şi c)

```
@echo off
echo Se creaza structura arborescenta de directoare ceruta de
echo enunt
echo Se folosesc comenzile md (made directory) si
echo cd(change directory)
echo comanda tree afiseaza structura arborescenta de directoare
for %%v in (test1, test2) do md %%v
cd test1
md sdir1
cd sdir1
md sdir2
cd ..\..
cd test2
md sdir1
cd sdir1
md sdir2
md sdir3
cd ..\..
tree > arb.txt
echo struct1...
type arb.txt
```

```
cd test2\sdir1
rd sdir3
cd ..\..
tree > tree.txt
echo struct2...
type tree.txt | more
cd test2\sdir1
rd sdir2
cd ..
rd sdir1
cd ..
tree > tree2.txt
echo struct3...
type tree2.txt
```

d)

```
@echo off
echo dir cu parametrul nume
echo generic de fisier
dir *.txt > texte.txt
type texte.txt | more
echo dir cu nume specificat
echo de fisier
dir %1
dir %1 | find "file(s)" >
fis.txt
type fis.txt | more
```

```
echo dir cu parametrii
echo specifici
dir /o:s > a.txt
type a.txt | more
dir /o:-s > a.txt
type a.txt | more
dir /o:d > a.txt
type a.txt | more
dir *.pas > pasc.txt
type pasc.txt| more
echo etc...
```

2. Concatenare

a) Să se creeze subarborele de directoare dat de parametri din linia de comandă.

b) Să se scrie un fişier `CONCAT.BAT` care concatenează fişierele date ca parametri în ultimul parametru.

Scanned with OKEN Scanner

c) Să se afişeze parametrii liniei de comandă.
d) Să se păstreze rezultatul de la punctul c) într-un fişier care va fi afişat pagină cu pagină.

Soluţie:

a)

```
@echo off
echo primul parametru va fi numele directorului parinte
echo parametrii urmatori corespund celor trei subdirectoare
echo fiu pt %1

md %1
cd %1
for %%v in (%2,%3,%4) do md %%v
tree > tree.txt
type tree.txt | more
del tree.txt
echo sterg acum structura arborescenta creata pt ca la un apel
echo sa pot folosi aceeasi parametrii...
for %%v in (%2,%3,%4) do rd %%v
cd ..
rd %1
```

b)

```
@echo off
echo se concateneaza fisierele parametrii (mai putin ultimul)
echo in linia de comanda si depune rezultatu in fisierul
echo precizat de ultimul parametru specificat la apelul
echo fisiserului *.bat

rem fixam numarul de fisiere de concatenat la 5
rem vor fi astfel 6 parametrii la executia bat-ului
del %6
for %%v in (%1,%2,%3,%4,%5) do type %%v >> %6
type %6 | more
```

Mai scurt, punctul b) se poate rezolva astfel:

```
@echo off
echo executa concatenare prin copierea din mai multe surse
echo in aceeasi destinatie (%6)
del %6
copy %1+%2+%3+%4+%5 %6
type %6 | more
```

c) şi d)

```
@echo off
echo afisez parametrii liniei de comanda
echo afisez parametru cu parametru pina cind gasesc
echo un parametru sir_vid ( dupa ultimul)
```

Scanned with OKEN Scanner

```
echo deplasarea in lista de param.ai liniei de comanda se face
cu comanda shift
del par.txt
:et
  if "%1"=="" goto sf
  echo %1 >> par.txt
  echo %1
  shift
  goto et
:sf
  type par.txt | more
  echo bye...
```

3. *Testare şi compilare*

a) Să se realizeze un fişier de comenzi `TEST.BAT` care testează dacă primul parametru e numele unui fişier din directorul curent şi testează dacă al doilea parametru este numele unui director (subdirector) din directorul curent. În cazul unui răspuns afirmativ, să se copieze fişierul în subdirector, altfel să se afişeze mesaje de atenţionare.

b) Să se realizeze un fişier de comenzi `COMA.BAT` care testează dacă cei doi parametri din linia de comandă permit executarea unei comenzi asupra unui fişier, adică:

`>COMA.BAT` *param1 param2* (*)

Dacă *param1* sau *param2* sunt şiruri vide (practic nu există) atunci să se afişeze că bat-ul trebuie folosit sub forma (*) unde:

```
param1 - este de tip fisier
param2 - operatie permisa pentru fisiere
```

Dacă se respectă formatul liniei de comandă, atunci `COMA.BAT` va executa comanda *param2* asupra fişierului cu numele specificat în *param1*.

Soluţie:

a)

```
@echo off
echo testez existenta fisierelor date ca parametru
echo pt %2 verific existenta intrarii in director (%2\nul)
if not exist %1 goto sf
if not exist %2\nul goto et
copy %1 %2
goto end
:et
  echo creez directorul cu numele specificat de %2
  md %2
  copy %1 %2
  goto end
:sf
  echo nu exista fisierele parametru in dir_crt
:end
```

Scanned with OKEN Scanner

b)
```
@echo off
rem dir %1 > nul
if "%1"=="" goto mes
if "%2"=="" goto mes
:ok
  echo executa comanda (%2) asupra fisierului (%1)
  %2 %1
  goto end
:mes
  echo folositi :
  echo %0 param1 param2
  echo param1 - tip fisier
  echo param2 - comanda de executat asupra fisierului (type, del)
  goto end
:end
  echo bye...
```

4. Compilări şi arhivări

a) Să se scrie un fişier de comenzi `COMPILE.BAT` care compilează toate programele Pascal şi C (`.cpp`) existente în directorul curent folosind compilatoarele `tpc` şi `bcc`. Rezultatele compilării ulterior se vor şterge.
b) Se vor arhiva programele Pascal care conţin în numele lor şirul `'poli'` (fac operaţii cu polinoame) într-o arhivă, respectiv programele C (`.cpp`) care au fost realizate în luna curentă şi au dimensiuni mai mari decât 100KB.

Soluţie

```
@echo off
rem la executie : compile.bat <numardelinii>
rem apelez compilatorul pascal pt sursele *.pas din dir_curent
for %%v in (*.pas) do tpc %%v >> pas.txt
rem compilez sursele *.c,*.cpp din dir_curent
for %%v in (*.c,*.cpp) do bcc %%v >> cpp.txt
rem tiparesc rezultatele compilarii, apoi sterg
rem fisierele auxiliare
type pas.txt | more
del pas.txt
type cpp.txt | more
del cpp.txt
rem sterg vechile arhive de programe *.pas si *.cpp
del arhiva*.arj
rem caut in directorul curent toate fisierele *.pas care contin
rem in nume subsirul "poli" (se afiseaza liniile returnate de
rem dir... nu numai numele lor)
rem parametrul /I pt comanda find face cautare neglijind
```

Scanned with OKEN Scanner

```
rem diferentele dintre literele mari si literele mici
rem corespunzatoare
dir *.pas | find /I "poli" > fis.txt
rem aplicatia facsir.exe construieste din fis.txt fisierul
rem temp.txt care va contine numele tuturor fis. pascal cautate
call facsir.exe PAS
rem apelez arhivatorul pt lista de fisiere din temp.txt
rem ( se arhiveaza in arhiva.arj toate fisierele de forma
rem *poli*.pas )
arj a arhivap !temp.txt
rem sortez fisierele *.cpp din dir_curent dupa data crearii si
rem dupa marime
rem arhivez fisierele al caror nume e continut in temp.txt
type fis.txt | more
rem sortez fisierele *.cpp din dir_curent dupa data crearii si
dupa marime
dir *.cpp /O:D /O:-S | find /I "cpp" > fis.txt
type fis.txt | more
rem contruiesc cu facsir.exe fisierul temp.txt care va contine
rem numai numele fisierelor *.cpp din directorul curent (fara
rem alte informatii)
call facsir.exe CPP
tpc linii.pas
rem apelez linii.exe care copiaza din temp.txt numai %1 linii
rem in final.txt
call linii.exe %1
rem arhivez fisierele din lista continuta de final.txt
arj a arhivac !final.txt
type final.txt | more
```

Fişierul LINII.PAS care trebuie compilat:

```
var s:string;
    f,f1:Text;
    i,nr,code:Integer;
Begin
  val(paramstr(1),nr,code);
  Assign(f,'temp.txt'); Reset(f);
  Assign(f1,'final.txt'); Rewrite(f1);
  for i:=1 to nr do
  begin Readln(f,s); Writeln(f1,s) end;
  Close(f1); Close(f)
End.
```

Fişierul FACSIR.PAS care trebuie compilat:

Scanned with OKEN Scanner

```
var f,rez:Text;
    s,sir,ext:string;
    c:Char;
    i,j,p:Integer;
Begin
  Assign(f,'fis.txt'); Reset(f);
  Assign(rez,'temp.txt'); Rewrite(rez);
  i:=0;
  if paramstr(1)='PAS' then ext:='PAS';
  if paramstr(1)='CPP' then ext:='CPP';
  while not eof(f) do
  begin
    sir:='';
    i:=i+1;
    Readln(f,s);
    p:=Pos(' ',s);
    for j:=1 to p-1 do sir:=sir+s[j];
    sir:=sir+'.'+ext;
    Writeln(rez,sir);
    Writeln('sirul ',i,' este :',sir)
  end;
  Close(f); Close(rez)
End.
```

5. *Concurs de programare*

Fie un concurs de programare. Se consideră `n` fişiere (`n` cunoscut sub formă de constantă) `T1,T2,...,Tn` care reprezintă fişiere de intrare pentru o problemă de concurs.

Să se scrie un fişier de comenzi `RUNTEST.BAT` care să realizeze:

a) copierea programului executabil al concurentului din directorul `C:\CONCURS` în directorul curent; nu se va afişa dialogul;

b) lansarea fişierului `RUNPROG.BAT` care are funcţiile specificate mai jos:

b1) copierea fişierelor `Ti` în directorul `C:\CONCURS` şi redenumirea lor în `INPUT.TXT`; nu se va afişa dialogul;

b2) lansarea în execuţie a executabilului concurentului (`conc.exe`);

b3) redenumirea fişierului de ieşire a concurentului din `OUTPUT.TXT` în `Ci`; nu se va afişa dialogul;

c) lansează în execuţie programul executabil `eval.exe` care verifică conţinutul fiecărui fişier de ieşire `Ci`;

d) şterge conţinutul directorului `C:\CONCURS`.

Scanned with OKEN Scanner

Soluţie:

```
RUNTEST.BAT
@ echo off
copy C:\CONCURS\conc.exe>nul
for %%i in ( 1 2 3 4 5 6) do call runprog %%i
eval.exe
```

```
RUNPROG.BAT
@echo Voi rula programul pentru testul numarul %1
copy C:\DATE\t%1. input.txt>nul
conc.exe
copy output.txt c%1.>nul
```

6. *Tăiere şi reunificare*

Să se scrie un fişier de comenzi care să permită:

a) lansarea NC;

b) căutarea într-un director precizat a unui fişier (directorul şi numele fişierului vor fi date ca parametri în linia de comandă);

c) în cazul în care fişierul se găseşte în director să se împartă în 10 fişiere; altfel să se afişeze un mesaj: `'ESEC LA CAUTARE'`). Să se copieze cele 10 fişiere într-un director creat pe loc, care va avea numele fişierului dat (fără extensie);

d) să se reunifice primele `t` fişiere (`t` va fi parametru în linia de comandă) din directorul astfel creat în fişierul `NOU.TXT`.

```
@echo off
echo testam existenta directorului dat de %1
if not exist %1\nul goto et1
echo daca directorul exista, cautam fisierul care trebuie
divizat
cd %1
if not exist %2 goto et1
goto et2
:et1
  echo esec la cautare
  goto sf
:et2
  echo am gasit
  nc
  echo ok !
rem la iesirea din nc (cu F10) se va afisa mesajul specificat
rem de echo..
  goto sf
:sf
  echo bye
  cd ..
```

Scanned with OKEN Scanner

Culegerea conține:

- probleme de algoritmică, majoritatea rezolvate în Pascal
- probleme de gestiune a bazelor de date (FOX-PRO)
- probleme care verifică îndemânarea de a lucra în mediul Windows și cu editorul Microsoft Word
- probleme de verificare a cunoștințelor aferente sistemului de operare DOS

Această culegere constituie un sprijin pentru colectivele de profesori și pentru elevii care doresc să pregătească probleme pentru atestatul de informatică dar poate fi folosită și în scopul pregătirii pe parcursul celor patru ani de liceu.

Culegerea a fost realizată de un număr mare de profesori cu o bogată experiență în predarea informaticii la toate nivelurile. Astfel, culegerea conține o varietate mare de enunțuri de probleme rezolvate, abordând diferite metode și stiluri de programare.

• • •

Dacă vreți, considerați că participați la o întrecere și încercați să rezolvați aceste probleme (suntem convinși că puteți) mai elegant și cu o eficiență în programare mai ridicată.

ISBN 973-97515-2-0

Lei 25000

Scanned with OKEN Scanner